IT'S HERE!

Case studies and survey about the progressive JavaScript framework for developers and CTOs

State of Vue.js 2019

Brought to you by Monterail
In collaboration with Evan You and Chris Fritz

What's Inside?

Preface

Vue.js has brought us business worth around €1M since we first adopted it in 2016.

But financial stability isn't the only thing that Vue has helped us with: happiness among our employees has soared and so has our sense of community, while our clients are astonished with the speed of development and the quality of the end results—all of which are priceless for a software development company like Monterail. And we owe it all to Vue.

The first State of Vue.js report was downloaded over 8,000 times by people from all around the world. The NPS (net promoter score) from that edition equaled 8.3 which is an amazing score for a book like that. It was apparent that such a report was needed and we thought it would be a good idea to develop it further and updated from time to time.

Like its predecessor, this new, revised edition of the report was created for three main reasons. One, it was to serve as a reliable source of Vue.js business use cases to anyone interested in getting a sneak peek of how other companies use Vue.js. Two, it was supposed to reach individuals who have never heard of Vue and provide them with good reasons to give the framework a closer look. Three, with the report at our disposal, we'd never again have to convince our clients that Vue.js is a ready-to-use solution that has everything we need to build all kinds of applications.

But things have changed for Vue and it's become much more stable and comprehensive than it has been in 2017. Plus, we believe that start-ups and SMBs have finally realized the power of Vue.js and understand the value it brings. So we made it our mission to send the same message to enterprise-level organizations, as we know that some of them have already adopted the framework to great success—hence the IBM case study featured in this updated edition.

The Rationale Behind the State of Vue.js Report

With Vue in our tech stack we can efficiently deliver better products—it helps us drive our business and make clients happy, and so we believe it deserves all attention and love. With that in mind, we embarked on the journey to evangelize to developers and businesses and spread the word about Vue. That's how we ended up curating the weekly Vue-newsletter and organizing VueConf, the first international Vue.js conference in the world.

The report you're reading is yet another milestone in that mission. It was created for three primary reasons. One, to provide a reliable source of Vue.js business use cases so anyone can get a sneak peek at how other companies use Vue.js. Two, to reach more individuals who have never heard of Vue and provide them with good reasons to give the framework a closer look. Three, to never, ever again have to convince our clients that Vue.js is a ready-to-use solution and has everything we need to build all kinds of applications.

Contents of the Report

The State of Vue.js report offers a business owner's and developer's perspective on Vue. We surveyed over 1,500 specialists from all around the world to find out their experiences with Vue, and what they like and dislike about the framework the most. We also asked the companies featured in the 2017 edition to provide an update on their journey with Vue.js, and included four new case studies. Just like a year ago, we also included a sneak peek of what's coming next for Vue penned by Evan You, the creator of the framework himself.

Thanks for taking the time to read through the report,

Joanna Staromiejska
Content Specialist
at Monterail

Karolina Gawron
Content Marketing
Manager
at Monterail

Marta Klimowicz
Head of Marketing
at Monterail

Szymon Korzeniowski
Head of Frontend
at Monterail

Contributors

This report features a curated base of knowledge and experiences collected from many important industry figures and Vue.js evangelists. We wouldn't be able to pull this off if it wasn't for them. Thank you for your help, support, and kindness.

Evan You
Creator of Vue.js

Chris Fritz
Vue.js team core member

Sylvain Simao
Technical Lead at Clemenger BBDO Melbourne

Clement Ho
Frontend Engineering Manager at GitLab

Thibaut Davoult
Growth Engineer at Livestorm

Andrew Courtice
Lead front-end developer at Fathom

Stephane Rodet
Senior UX Engineer at IBM Design

Yuriy Nemtsov
Software Engineer & Manager at Behance

Taylor Otwell
Founder of Laravel

How Developers Use Vue.js?

Like its predecessor, the 2017 State of Vue report, this new, updated version was also supposed to allow us to learn more about the community of professionals using Vue.js framework. Using an online survey, we sourced data from both software developers and Chief Technology Officers which we then examined to gain the insights on:

- the popularity of Vue.js in their organizations,
- the reasons behind adding Vue to their tech stack and the doubts that accompanied the decision,
- the solutions they choose when developing Vue projects,
- the libraries/frameworks they use for frontend development,
- languages used for backend development,
- their predictions (and wishes) for the future of Vue.js.

Report Data

All data used to draft the report was collected in a survey conducted over a five-week period in November and December of 2018. We received 1,553 responses, mainly from software developers and Chief Technology Officers (88% of the respondents held one of these roles) whose organizations currently use Vue.

We also asked Evan You, Vue creator, and Szymon Korzeniowski, Head of Development at Monterail, to comment on the results of this survey, to give us even more insights and a better understanding of the broader context.

We also asked Evan You, Vue creator and Chris Fritz, Vue Core Team member to comment on some of the survey results, in order to provide additional insights or to share their broader perspective.

Key Insights

92% respondents would use Vue.js again for their next project

94% of the survey participants used the official documentation to learn about Vue

75% of the respondents pointed to ease of integration as the biggest advantage of Vue.js

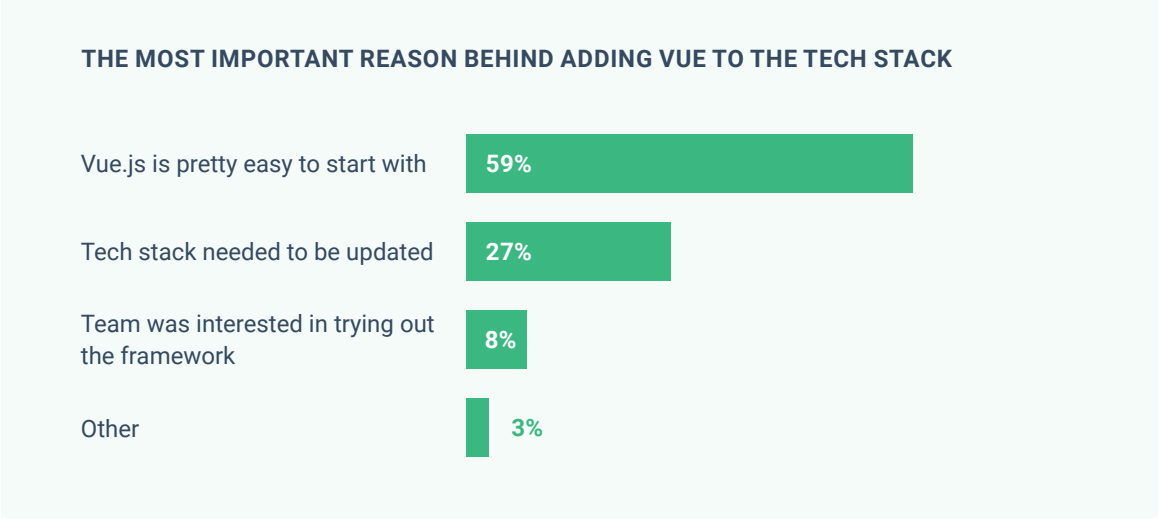
60% of the survey participants believe Vue.js will become even more popular within their organizations in the next 12 months

58% decided to add Vue to their tech stack because it's a pretty easy framework to start with

Survey Questions

What was the most important reason behind adding Vue.js to the technology stack?

More than a half of the respondents describes Vue.js as easy to start with. Start-ups choose it to enable faster MVP development, while in enterprise companies its adoption is usually driven by its its ease of integration with existing CMSes.



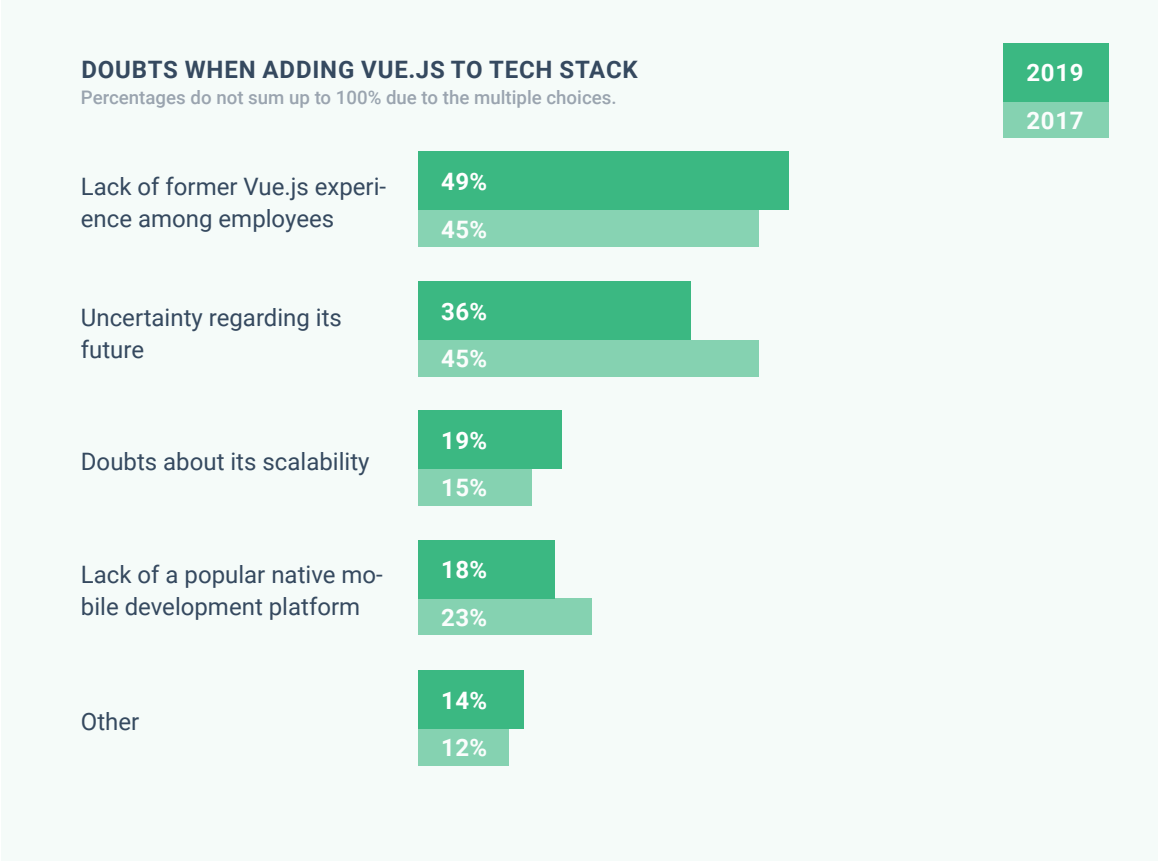
It should be noted that in our **2017 survey, a similar portion (59%) of the respondents chose the same reason behind adding Vue.js to their technology stack**, implying that key adoption drivers have remained mostly unchanged between then and now.

What were the doubts you and your team had when planning to add Vue.js to your tech stack?

Almost 50% of the respondents say that lack of former Vue-related experience was their main doubt when planning to add Vue.js to the tech stack.

This number is pretty similar to the one we saw in 2017 when the answer was picked by 45% of the people we polled.

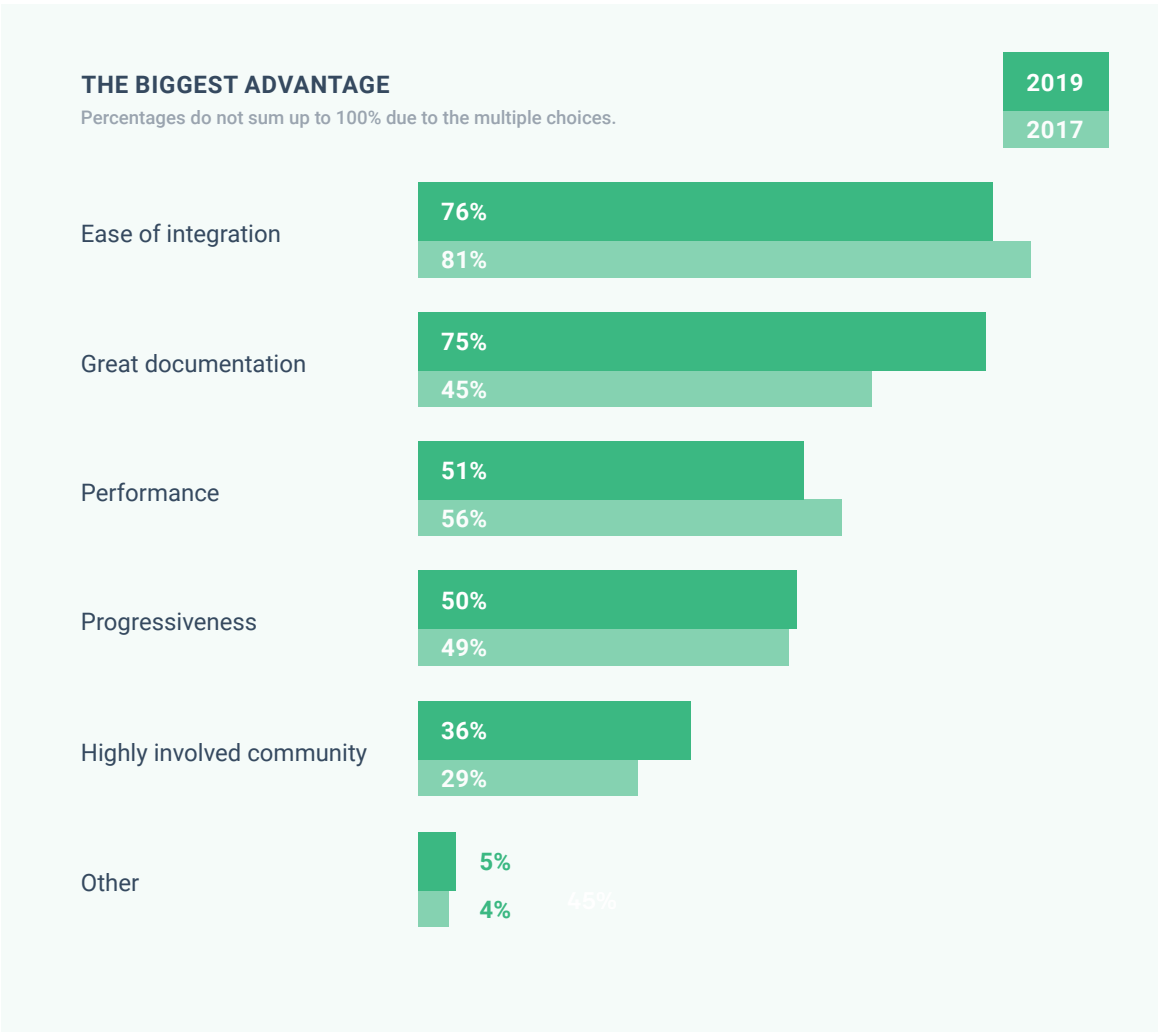
It bears noting, however, that in the span of nearly fifteen months that separated the two survey, trust in the future of the framework grew by nearly 10%—where in 2017 45% of our respondents mentioned doubts about the future of Vue.js, similar reservations were brought up by only 36% of the 2019 respondents.



That's definitely a good sign, and we hope the number will go down further in the future as we put more work into the project's governance, contribution management and long term sustainability.

Evan You 

What are the biggest advantages that Vue.js brings to your organization?



Is there anything you're missing when it comes to Vue.js?

637 respondents (up from 481 in 2017) shared their suggestions through the survey. Again, we decided to pool these comments and requests in order to better flesh out the bigger picture.

Over 130 people pointed to Vue's lack of a mobile solution as one of its biggest flaws. This was also the top suggestion the last time we ran this survey.

> ***Evan: NativeScript's Vue integration is pretty solid now. Maybe not enough people are actually aware of it because it's not"official" - but I've only been hearing good things about it from those who have tried it. And Progress (the company behind NativeScript) is really doubling down on their investment in it. In the hybrid app space, Quasar has really matured and is close to 1.0, plus Ionic 4 is now fully compatible with Vue as well. So I believe we've got a solid list of solutions for anyone wants to build mobile apps with Vue today.***

50 replies mentioned the need for a bigger Vue ecosystem that would provide a better collection of tools and libraries.

46 of the respondents brought up the need for improving Vue documentation to enable smoother app development.

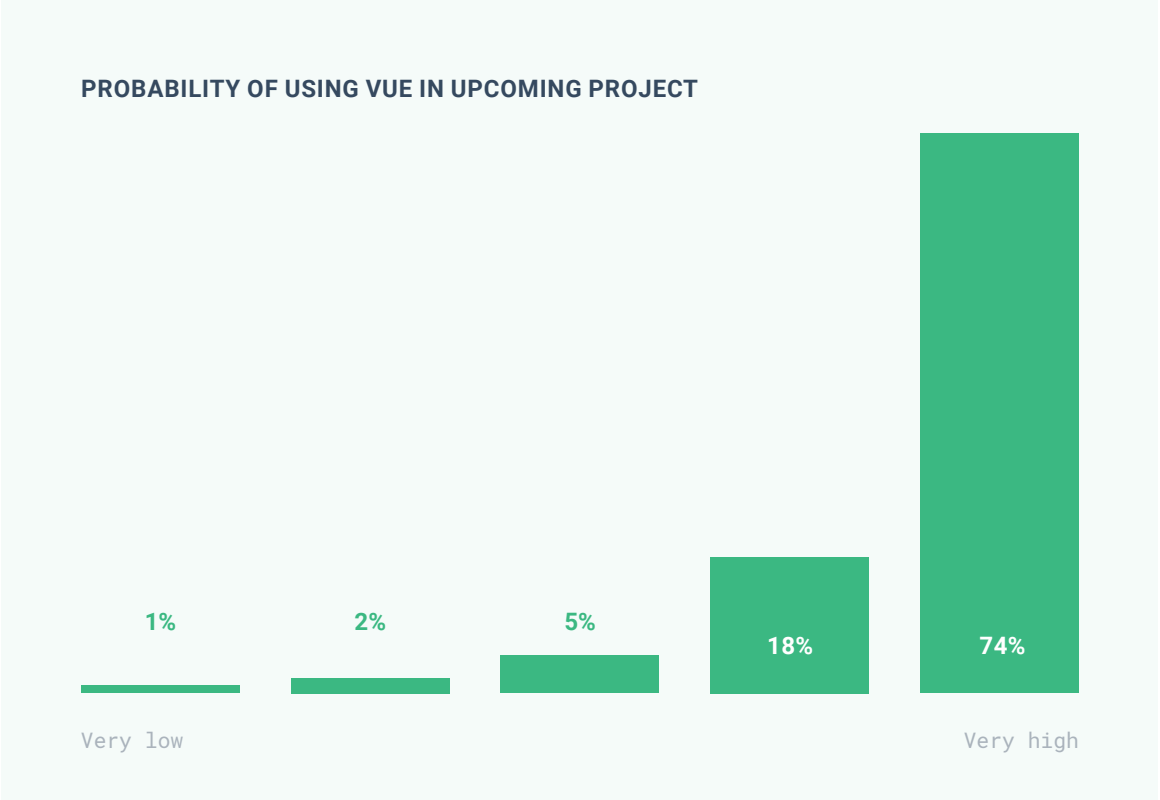
> ***Evan: I think our current docs has been serving us really well so far. But we do plan to revamp the docs together with 3.0—there's always room for improvements for sure.***

42 of the pooled replies mention better testing tools and libraries' need for further Vue development.

> ***Evan: 3.0 will likely have a simpler testing methodology compared to today's because the custom renderer API allows easy creation of custom renderers for testing purposes.***

35 people suggested growing and improving available learning resources to include even more real life examples, especially of implementation in enterprise apps. However, the last time we asked, this need was brought up by 67 people, so the decrease in this particular area seems to show that the number and quality of the available learning resources have improved since.

What's the probability of you or your company using Vue.js in upcoming projects?



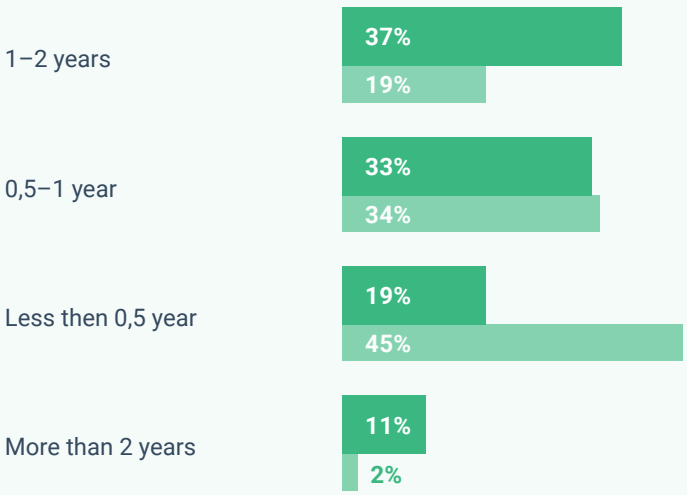
A whopping 92.3% of respondents claim there is a very high and high probability of them using Vue.js for their next project. A framework can ask for no better review than that. Having this many developers with prior experience in Vue.js who are still strongly motivated to use it is undoubtedly impressive.

How long Vue.js has been in use within your organization?

Last time we asked, 46% of the people we polled have been using Vue.js for less than six months. This particular metric saw the biggest shift in the intervening fifteen months. Today, **almost 37% of the respon-**

dents declare to have between one and two years of experience with Vue. This would indicate that those developers who started using Vue.js during or just before our last survey decided to stick with it.

HOW LONG VUE.JS HAS BEEN IN USE WITHIN YOUR ORGANIZATION?

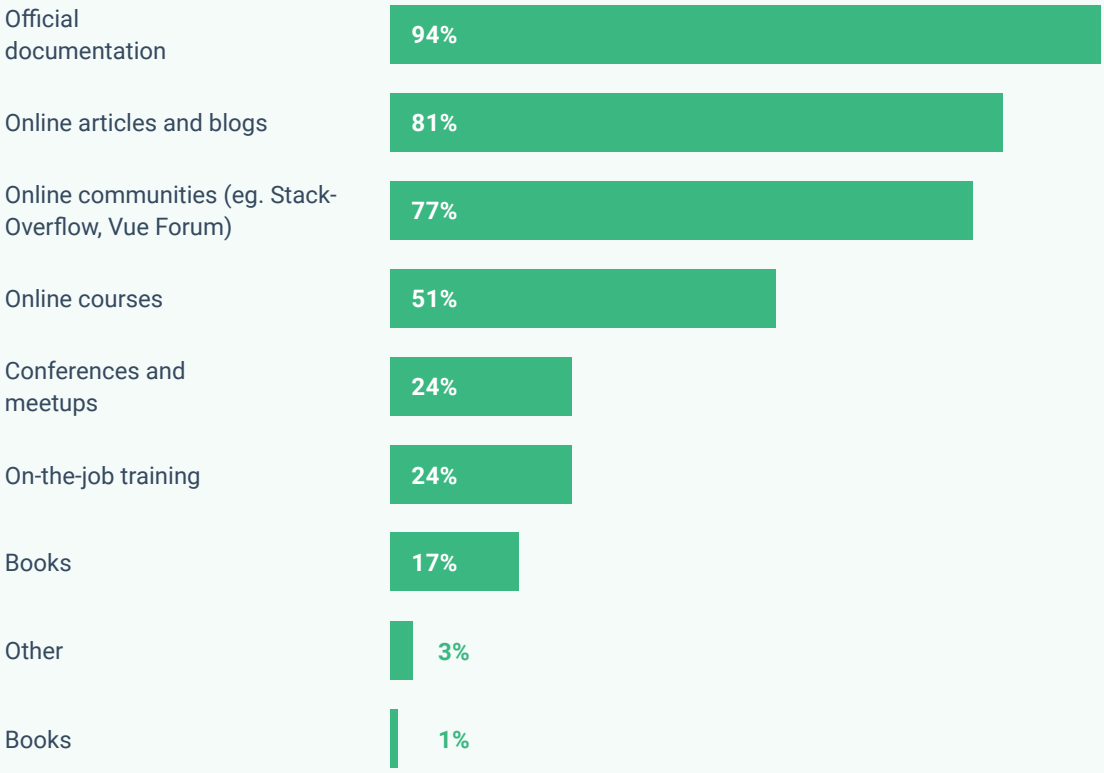


What resources do you use to learn about Vue.js?

Like two years prior, **the official documentation continues to be the most popular resource among those willing to improve their knowledge of the framework.** However, the popularity of online courses also saw a pretty significant increase, by over ten percentage points. The broader availability of more high-quality learning online courses may be one explanation for the change.

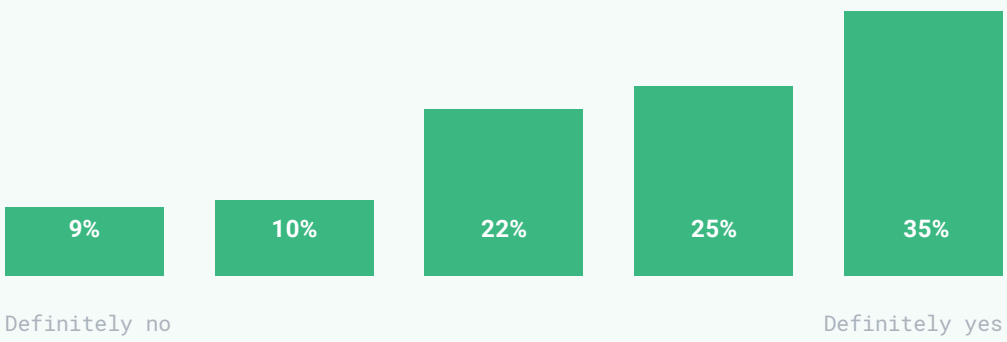
LEARNING RESOURCES

Percentages do not sum up to 100% due to the multiple choices.



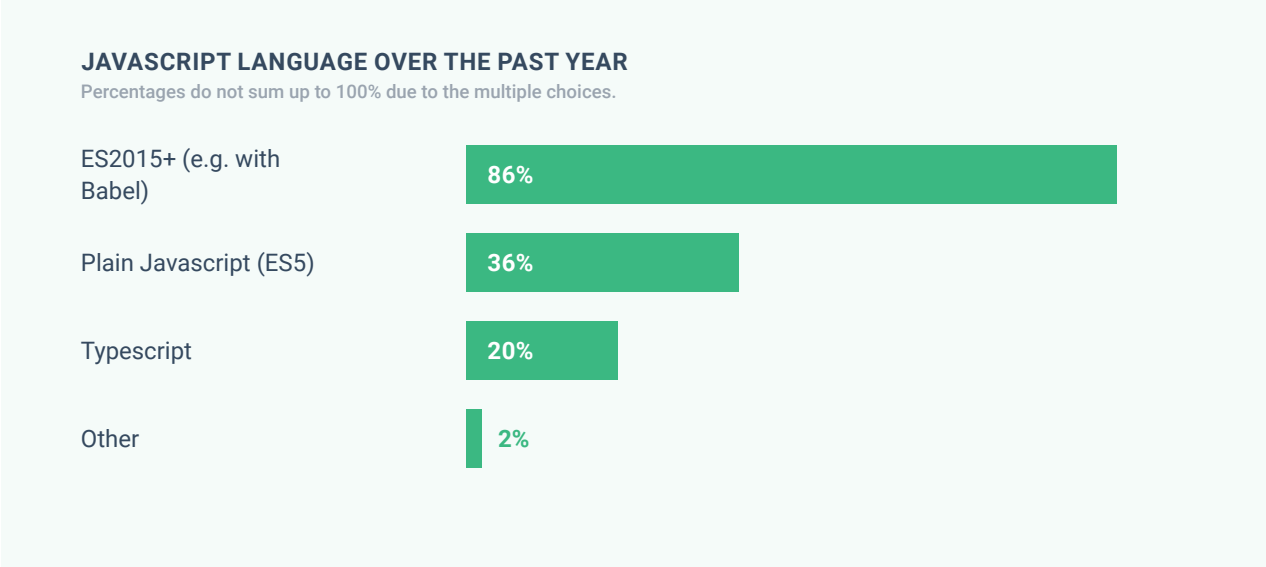
Do you think the number of employees using Vue.js in your organization will increase in the next 12 months?

INCREASE IN THE NUMBER OF EMPLOYEES USING VUE.JS



Over 59% of respondents are convinced Vue.js is going to get even more popular in their organization within the next 12 months. To compare: in 2017, the number was 54%. It is also worth stressing that among those respondents who work at large enterprises (with over 1,000 employees) over 73% are certain that Vue is going to be widely adapted inside their companies.

In Vue.js projects over the past year, which of the following have you used for JavaScript?

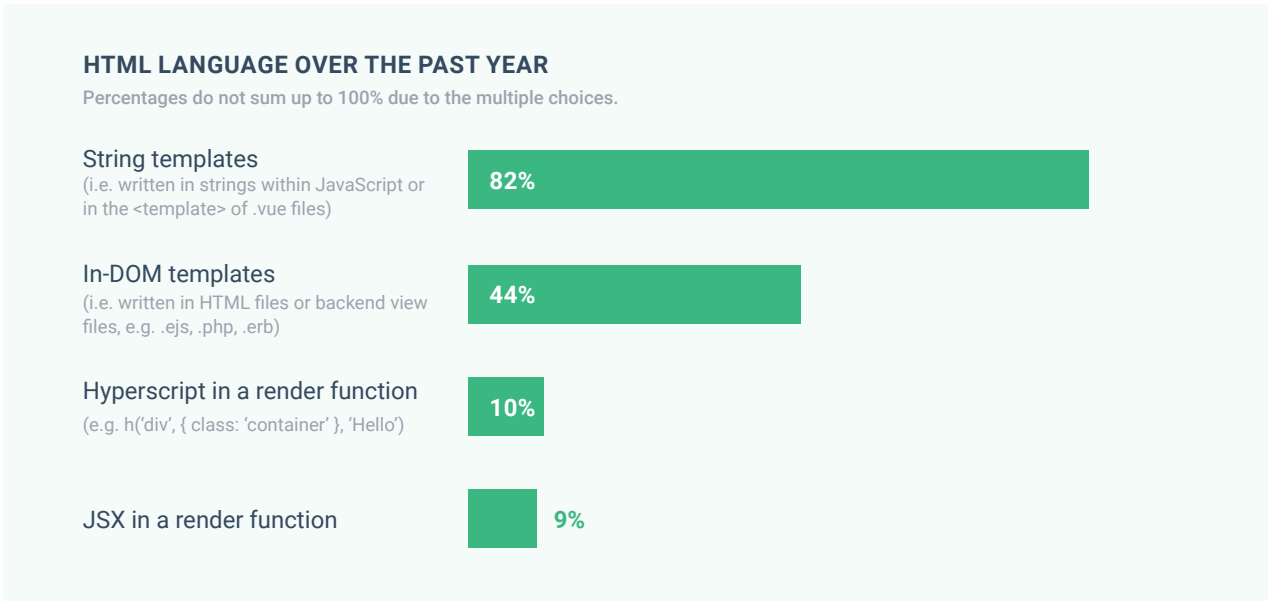


Szymon: With major new features introduced in ES2015 and newer versions, modern JavaScript has been embraced by developers—and browsers—everywhere. Tools like Babel, which can be easily added to a project thanks to Vue CLI, allow for effortless transpilation to ES5 for older browsers. Therefore, 86% respondents have been using modern JavaScript in their Vue projects, whereas 35.7% have worked with plain ES5.

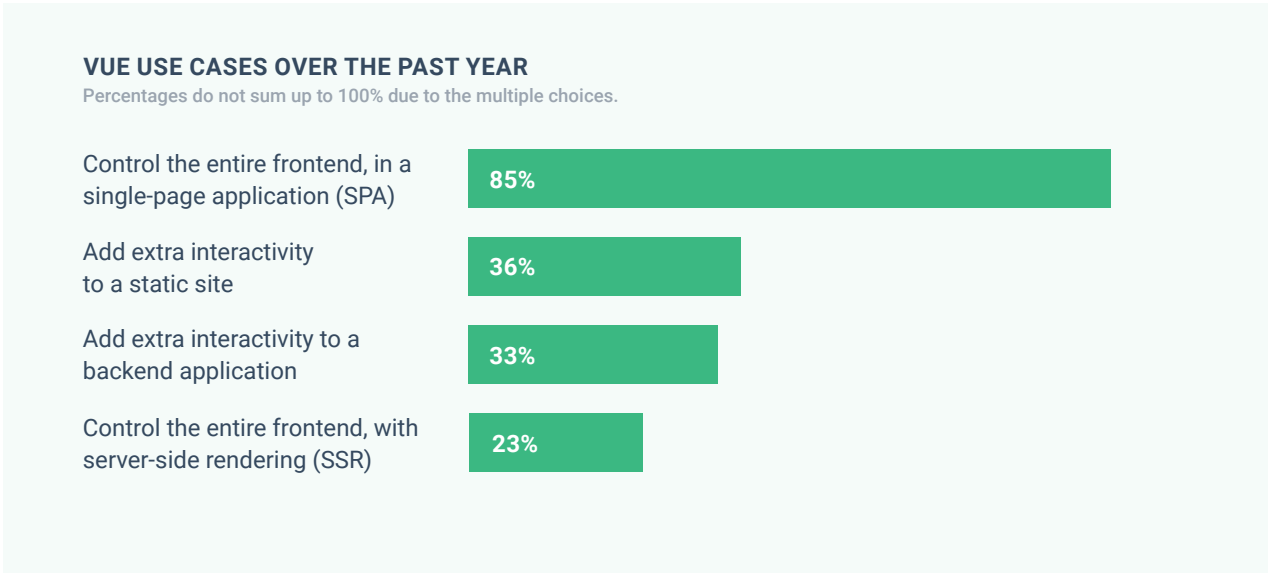
TypeScript is also fast becoming a popular option in the Vue ecosystem, with stronger support for the language introduced in Vue 2.5 and first-class sup-

> *port in Vue CLI. This number is likely to rise even further in the future, as Vue 3.0 will be rewritten from the ground up in TypeScript.*

In Vue projects over the past year, which of the following have you used to write HTML?

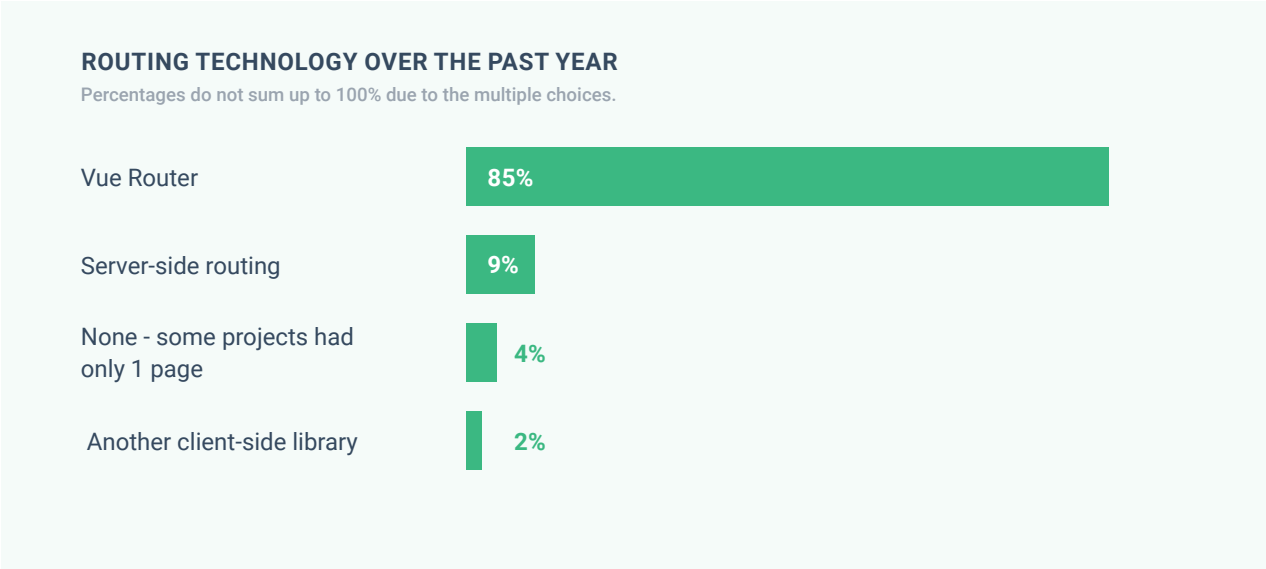


In Vue.js projects over the past year, which of the following use cases for Vue have you had?



Szymon: Vue's truly progressive nature, covering a broad spectrum of possible use cases, is reflected in the results of our survey. 85% of respondents have used Vue as a complete solution to build a full-blown single-page app, with 23.1% going even further using it together with server-side rendering. On the other hand, "drop-in" use cases for Vue—adding interactivity to existing static sites or background applications—are also common, with 36.5% and 32.6% respondents, respectively, using it this way.

In Vue projects over the past year, which of the following have you used for routing?

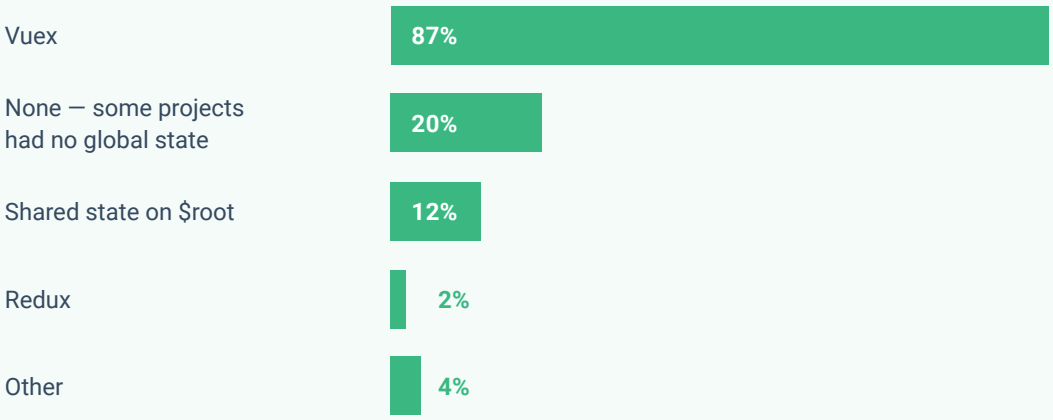


Szymon: As the official Vue routing library, Vue Router is a strong leader here, with nearly 85% respondents having used them in their projects over the past year. About 10% keep their routing on the server side, while 4% use no routing at all.

In Vue.js projects over the past year, which of the following have you used for global state management?

GLOBAL STATE MANAGEMENT OVER THE PAST YEAR

Percentages do not sum up to 100% due to the multiple choices.

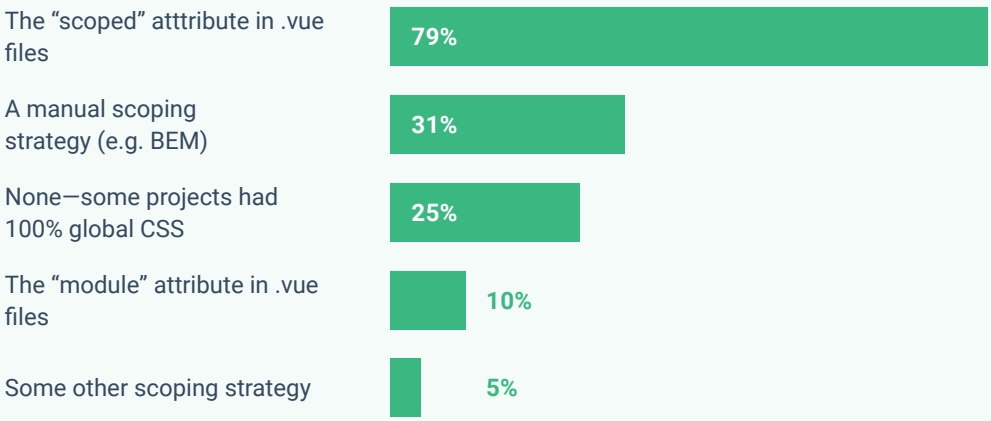


> *Szymon: Similarly to Vue Router, Vuex is the official state management solution in the Vue ecosystem, and is therefore used by the majority of respondents—nearly 87%, to be precise. Not all projects require complex state management, however—around 20% respondents have worked on projects without any global state, and around 11% have opted for using a shared state.*

In Vue.js projects over the past year, which of the following have you used to scope CSS?

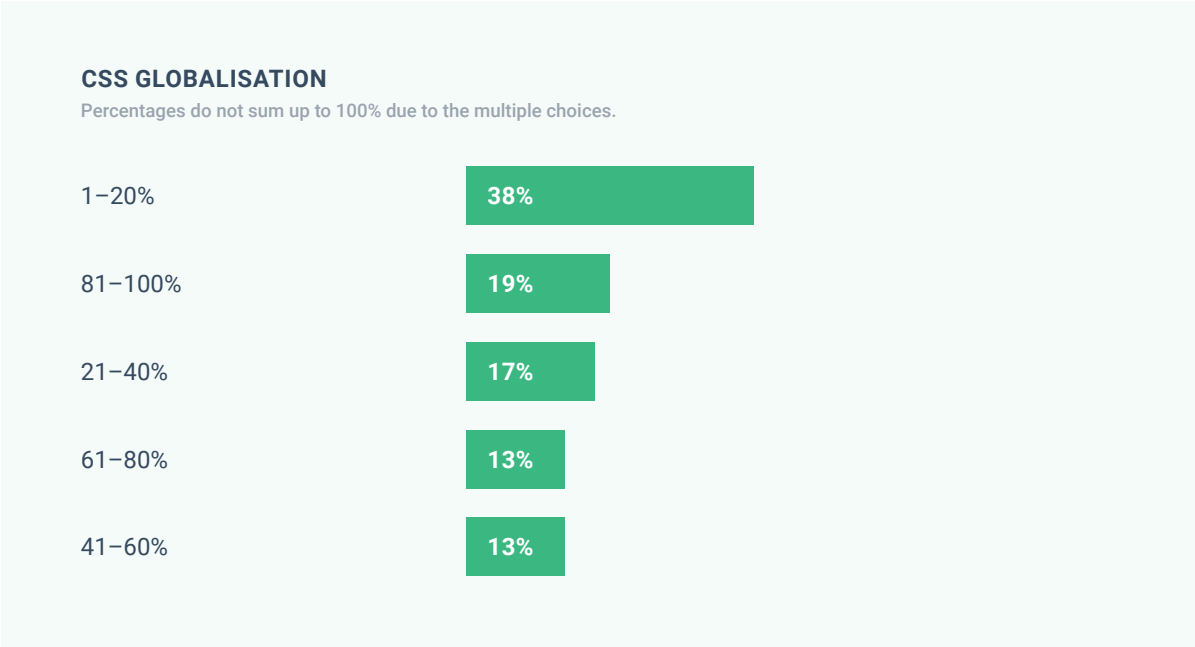
CSS SCOPING OVER THE PAST YEAR

Percentages do not sum up to 100% due to the multiple choices.



Szymon: The ability to create scoped styles in .vue files with a simple HTML attribute has proven to be a whopping success, with nearly 80% respondents using it to scope CSS. Only less than a third of developers are still using manual scoping strategies like BEM.

In your last Vue project, how much of the CSS was global?



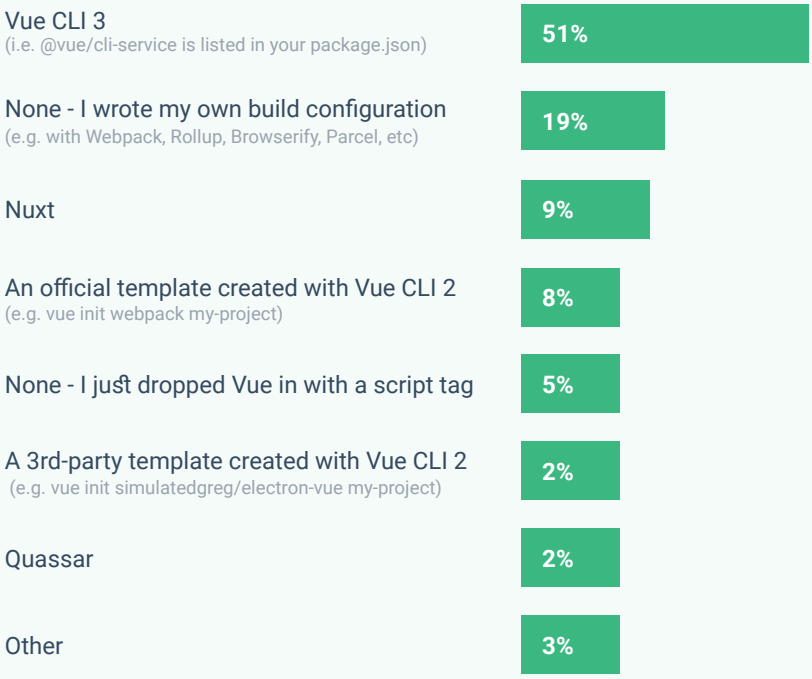
Szymon: Despite the popularity of scoped styles, global CSS is still a thing. Even though 38% of respondents use little to no global styles, for nearly 20% it comprised the majority of styles in their last project.

Which CLI tool did you use to create your last Vue project?

Szymon: 2018 saw the stable release of Vue CLI 3, which has made it super easy to kickstart new projects with a pre-configured Webpack 4 setup, multiple integrations, and an awesome graphical user interface. More than half of the respondents were quick to jump on the Vue CLI 3 bandwagon, using it to setup their most recent project.

CLI TOOLS

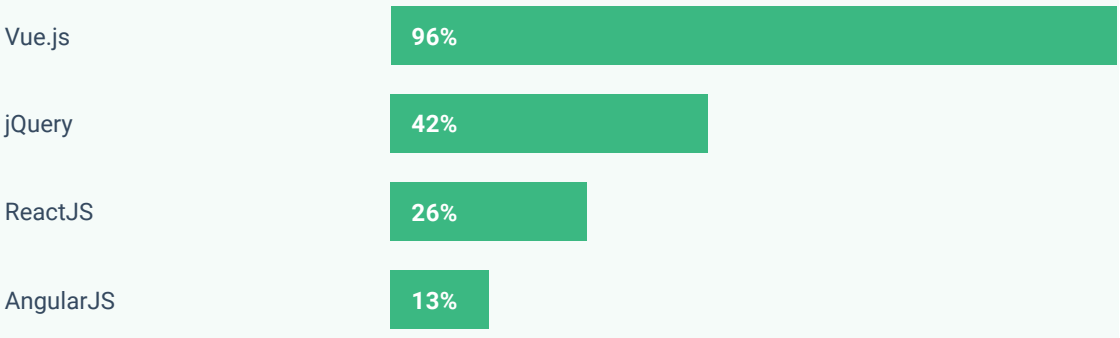
Percentages do not sum up to 100% due to the multiple choices.

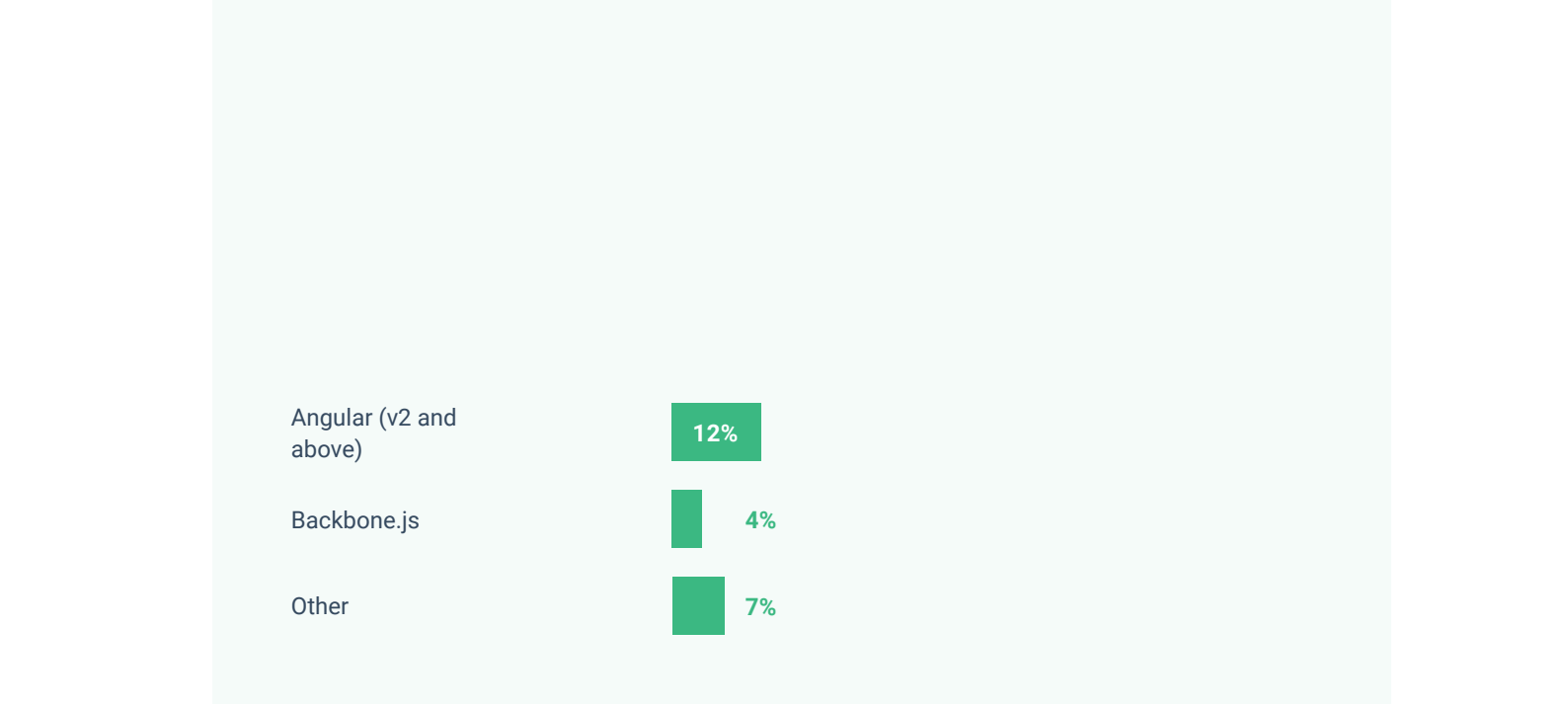


What libraries/frameworks do you use for frontend development?

LIBRARIES OR FRAMEWORKS

Percentages do not sum up to 100% due to the multiple choices.

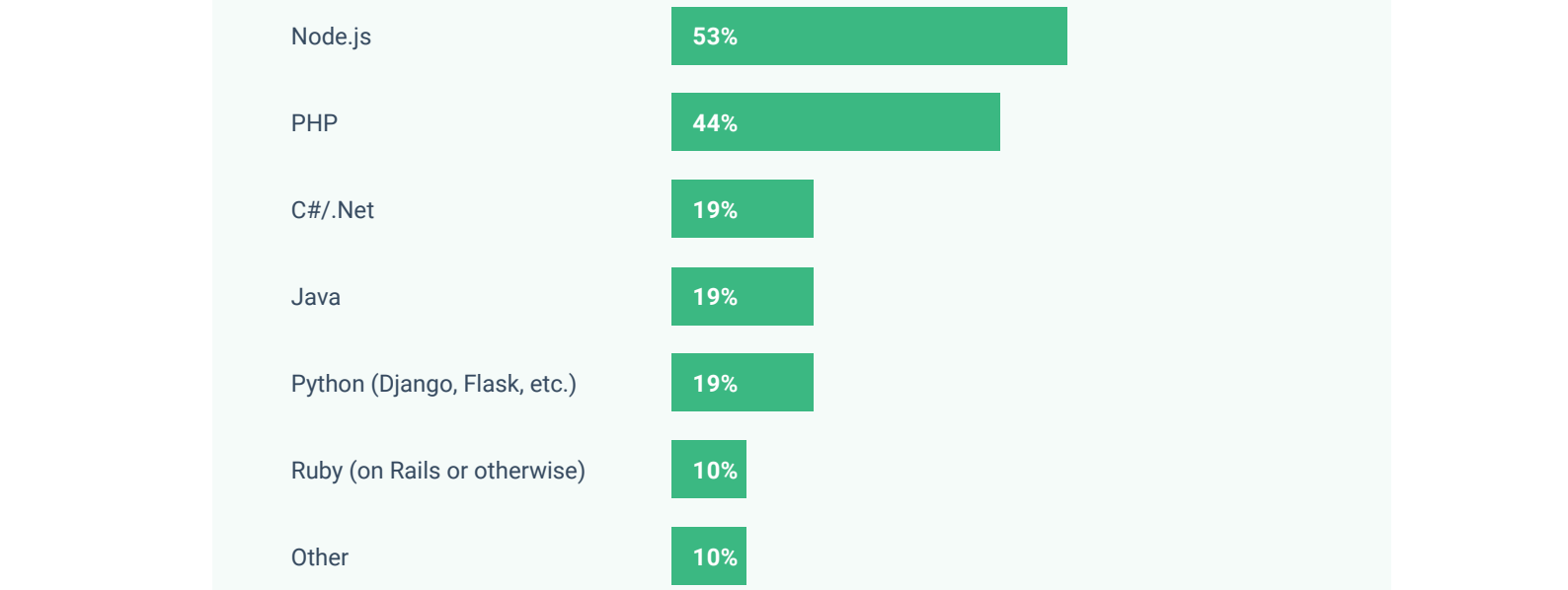




What languages does your organization use for backend development?

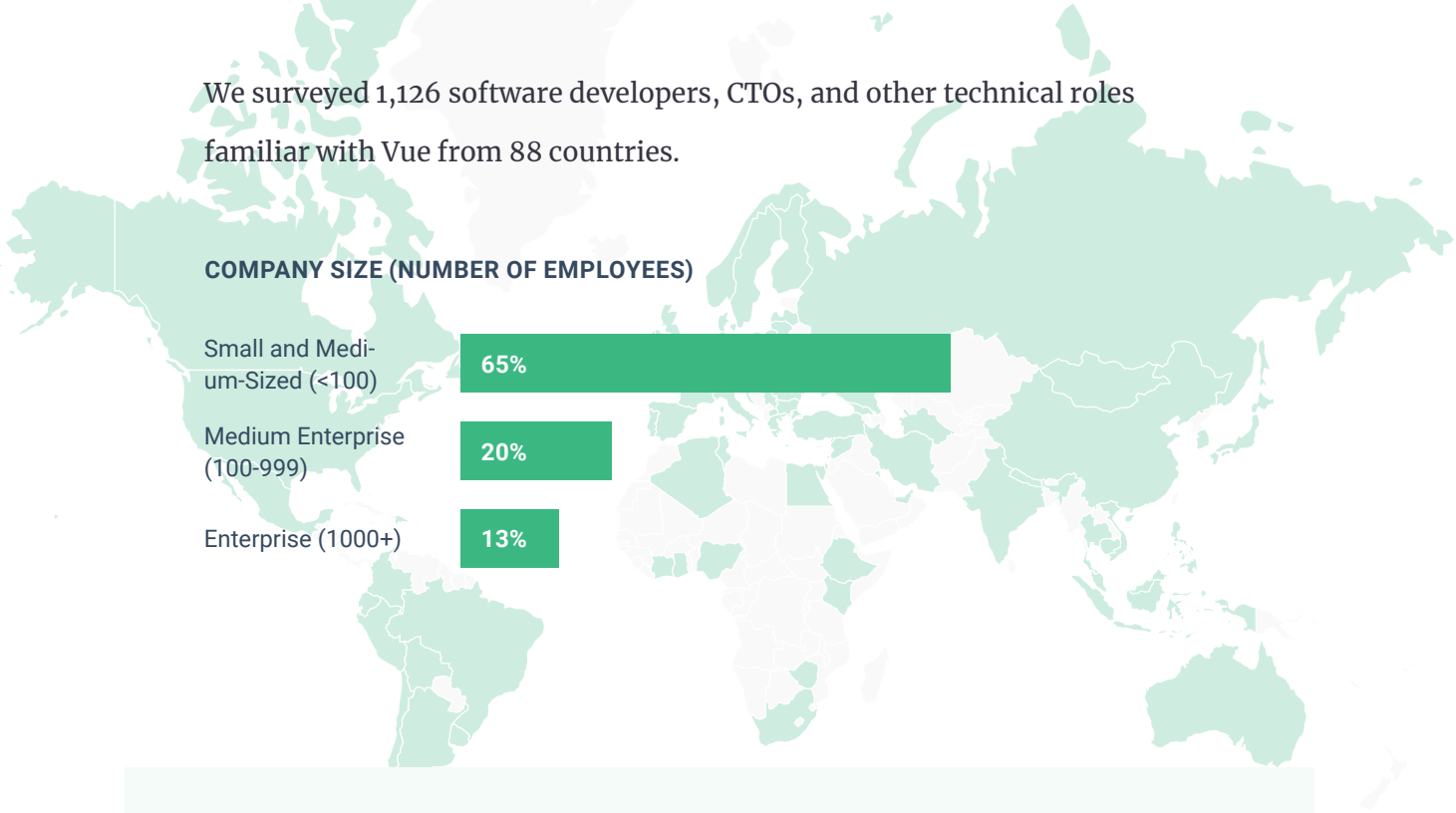
LANGUAGES FOR BACKEND DEVELOPMENT

Percentages do not sum up to 100% due to the multiple choices.



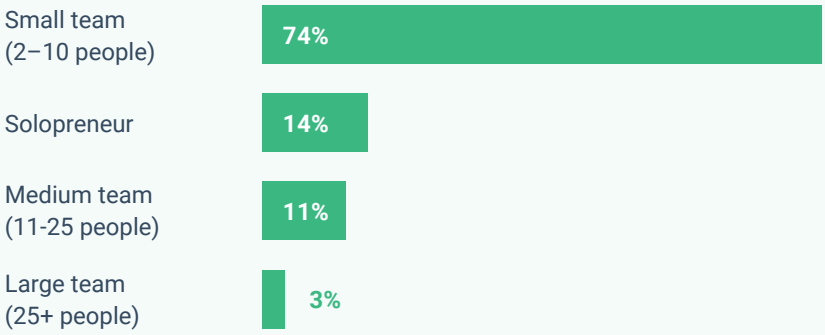
Demographics

We surveyed 1,126 software developers, CTOs, and other technical roles familiar with Vue from 88 countries.

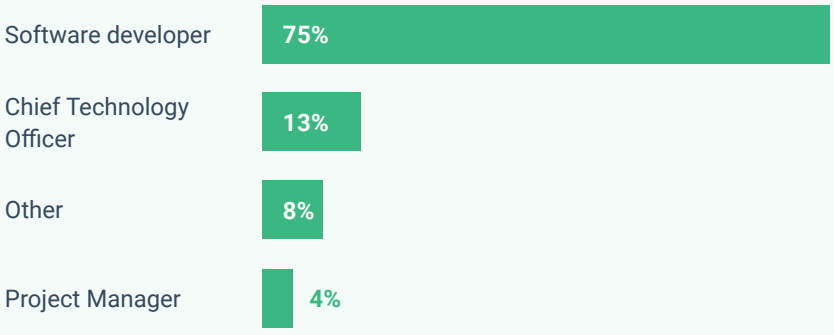


COMPANY SIZE (NUMBER OF EMPLOYEES)

TEAM SIZE (NUMBER OF TEAMMATES)



ROLE IN ORGANIZATION



One Year Later with Vue.js

You may be wondering what's going on with the companies we interviewed last year. Has Vue stood the test of time?

Behance, Clemenger BBDO, GitLab, and Livestorm all decided to share what was happening with their products and what's changed over the past year. We asked them the following four questions:

- Do you still use Vue.js in your products? Why/why not?
- What has changed in your usage of Vue.js?
- How many devs use Vue.js in your company now vs. a year ago?
- How did your company grow since we last spoke and how has Vue.js contributed to this?

Check out their case studies from 2017 and then dive into the second chapter of their stories. Hope you'll find them inspiring!

Behance

Yuriy Nemtsov
Software Engineer
& Manager
at Behance

Behance is the leading online platform for showcasing and discovering creative work.

Do you still use Vue.js in your products? Why/why not?

Yes, we do. And there are two reasons why I think Vue.js continues to be the best technology for Behance: SSR and ergonomics. We care about having a low TTI and wouldn't be able to achieve ours without Vue's superb SSR support. It's well thought out and helpfully documented. From the vue-loader, bundle renderer, full support in the vue-router and vuex, built-in component caching and even asset preload / preflight handling -- the efficiencies we gain from a single integrated SSR-aware ecosystem are unparalleled. And this isn't specific to SSR. Vue is built to be a progressive library where the independent components are designed to work together really well. They're ergonomic, and we love that about Vue.

What has changed in your usage of Vue.js?

On the state management side, we started with a Vuex store where all of our modules were registered at initialization time. Only later did we realize that for the size of our application, and specifically the fact that it is a website with independent sections, this approach did not scale well. Accessing any page of the site made the user download the Vuex

modules (and their supporting utilities and API communication libraries) for the entire application. A page that only needed a few KB of state management would be punished by ten times that with no benefit. What we do now is dynamically register Vuex modules when they are needed and unregister them when they aren't. In conjunction with async components and vue-router, we now only load the components and Vuex modules that are required for that specific page and dynamically import others when needed.

How many devs use Vue.js in your company now vs. a year ago?

We have ten people working on Vue.js full-time now. If memory serves, that's one or two people more than last year.

How did your company grow since we last spoke and how has Vue.js contributed to this?

Since last year, we've re-written our login homepage in Vue (a success, based on the overwhelmingly positive user feedback and the data we see). By the time you read this, we'll be done with two more major sections of Behance; all using efficient, server-side rendered Vue, allowing the people using Behance to showcase their work and be inspired by the beautiful work of others.

Clemenger BBDO Melbourne

Sylvain Simao
Technical Lead

Clemenger BBDO is a full service agency offering a full suite of capabilities including brand strategy, integrated creative development, CX, and digital services.

Do you still use Vue.js in your product? Why/why not?

Our interactive department act as creative firepower for Clemenger BBDO, and as such we need the ability to prototype highly interactive experiences at pace. As of today, Vue.js is still the one framework on the market that gives us the required velocity and flexibility to deliver on this type of work. Besides, our development team enjoys working with Vue, so I don't think we'll be moving away from it anytime soon.

What has changed in your usage of Vue.js?

The biggest game-changer for us probably happened after Vue CLI 3 has been announced. We've immediately adopted it into our workflow, even before the stable version was released. Vue CLI 3 helped improve our speed and consistency while reducing configuration fatigue and providing powerful instant prototyping tooling.

Another change worth mentioning is our cross-platform use of Vue.js. During the past year, we've started using the framework to create desktop apps in combination with Electron, and more recently to build a native mobile app using NativeScript-Vue.

How many devs use Vue.js in your company now vs. 2017?

In 2017, our entire developer team was already using Vue.js, and this hasn't changed except our team is now almost twice as large. Six developers are using Vue.js full-time now.

This coming year, we plan to go hard on voice, AI, IoT, and the experiential Web. Our technical expertise with Vue.js and its ecosystem will be decisive to support the projects ahead; this is why we empower our developers to experiment with all upcoming technology, including Vue-related resources and libraries.

How did your company grow since we last spoke and how has Vue.js contributed to this?

Interactivity is at the core of our best creative projects, making digital the fastest growing area of our agency. This year alone, we've launched some of our best campaigns and projects, using Vue.js as the central piece.

Among our most recent launches: NAB Yourself, a voice-activated experience for the National Australia Bank; Deadly Questions, a conversation platform between non-Aboriginal Victorians and the Victorian Aboriginal community; Snickers Hungerithm US, the stock exchange for Snickers chocolate bars; DrinkWise The Internet Remembers, a Web-based augmented reality experience that aims to stop young adults drinking too much.

Gitlab

Clement Ho
Frontend Engineering Manager

GitLab is an open core, integrated solution for the entire software development lifecycle.

Do you still use Vue.js in your product? Why/why not?

Yes! We are using more Vue this year than ever before. Vue.js continues to enable our frontend team to ship features quickly.

What has changed in your usage of Vue.js?

Our original vision for Vue at GitLab was to use it to help implement and maintain complex frontend features, since we were traditionally a large Rails + jQuery application and we didn't want to do a Vue rewrite for the sake of iterating quickly. Over the past year, we've started using Vue.js for majority of our features (including those that weren't as complex).

In addition, we are now even more fully committed to Vue.js. We created another project called gitlab-ui and have designated that as our Vue component library. Although the library is still pretty new, we've been able to set up a healthy ecosystem thanks to the tools provided by the Vue community. A lot of GitLab originates from Bootstrap, so it was very helpful to have Bootstrap Vue available as a dependency, as well as all the component library tools, such as Storybook, and their plugins. Storybook

plugins such as storyshots-puppeteer allow us to do visual regression tests on the component level. This enables us to improve the quality of our product and create a better user experience for our users.

How many devs use Vue.js in your company now vs. 2017?

We had 15 frontend engineers at GitLab in 2017, and now we have 29. We've effectively doubled the number of developers who use Vue.js at GitLab to help keep up with the product demands. We expect to double our frontend team headcount by the end of 2019.

How did your company grow since we last spoke and how has Vue.js contributed to this?

GitLab has had an amazing 2018. The company has grown a lot in size and it's great to see people understand GitLab as more than just source control. I would say that GitLab's investment in Vue.js is already paying off. Our involvement in the Vue.js community has helped us source great talent. Vue's fantastic documentation and guides also make it very easy for new team members to onboard quickly and start contributing code to our codebase.

Livestorm

Thibaut Davoult
Growth Engineer

Livestorm is a all-in-one web-based webinar solution. It helps companies like Workable, Pipedrive or Instapage do live sales demo or customer training.

Do you still use Vue.js in your product? Why/why not?

We still use Vue.js for all the frontend of Livestorm. That is true for every part of our Web conferencing software: the webinar and conference rooms and the user admin panel.

When we chose Vue in early 2016, our challenge was to build a stable MVP as fast as possible. After release day, with actual customers trusting Livestorm with their online events, our main task shifted to building for reliability and growth. We started in 2017, but the challenge of expanding to new users and new employees while keeping Livestorm reliable at all times was still very much the focus of 2018.

Vue keeps confirming to us that it's a great choice for scaling a team and product. That's why we've stuck with Vue and never even considered switching to an alternative. If anything, we're trying to push the use of Vue further by removing the (very marginal) remnants of external frameworks we've inherited from the early days, and making the most of the entire Vue.js toolbox.

What has changed in your usage of Vue.js?

In 2018, we kept working to make the most of what Vue.js has to offer. So the biggest changes were in how we structured the Livestorm app. We also made more use of state management via Vuex to rely even more on the store to handle data. We also kept pushing with components to separate each part of our app and make it easier to scale.

How many devs use Vue.js in your company now vs. 2017?

We started 2017 with 1 dev on Vue.js and ended with 2. In 2018 we had 3 devs focused on Vue.js.

How did your company grow since we last spoke and how has Vue.js contributed to this?

We've expanded from "just" live webinars to Web conferencing with Livestorm Meet—still 100% Web-based. We also added On-Demand Webinars, used by customers to automate the webinar process and let their events run by themselves. All these products are built on the same codebase and benefit from Vue.js components to avoid repeating our code.

In terms of customers, our year-on-year growth is above 300%, and we're still growing steadily.

We almost halved the number of support conversations per user between January 2018 and November 2018. Obviously a lot of things enter into account here and it's not all thanks to Vue, but the frontend definitely plays a big role in keeping our users happy, avoiding bugs and responding fast when they arise.

Case studies

Case studies are the best way to demonstrate the value of any tool, business, and... framework. Last year, we received dozens of emails mentioning case studies as a great selling point for Vue. They were proof that Vue was used commercially in multiple companies, and to great success, so we interviewed four more companies that were kind enough to share their experiences—Fathom, IBM and Laravel. Four very different businesses, all of which managed to successfully develop their products with Vue.js.

Note that when Evan sent a call out on Twitter for companies that would like to share their case studies with us, he received over 70 valid responses in total. And it was just one Tweet. Selecting just a few from the submissions was a difficult task, but those four we put down in this chapter seem to do Vue the most justice.

Fathom

Andrew Courtice
Lead front-end developer at Fathom

Fathom is a financial intelligence platform designed to integrate seamlessly with your existing accounting product and provide you with insightful analysis and reports.

Vue was a breath of fresh air after having used Angular in the past. I am convinced it offers the flexibility for any developer to create rich applications with ease. To anyone considering it at the moment—go for it and wait for the results, it's worth the investment

CHALLENGE

Difficulties with scaling due to code bundling

Outdated product design with fragmented styles

Duplicate code and inconsistent architecture

Difficult to debug and maintain

Use of old or poorly supported libraries causing a significant business risk

SOLUTION

Incrementally transitioning legacy features across and writing new features into new architecture

Creating a proof of concept

Putting a demo together and presenting it to management

OUTCOME

Codebase easy to maintain, test, and share across multiple applications

The app's memory usage reduced by half

The stylesheet cut down by 67.3%

Challenge

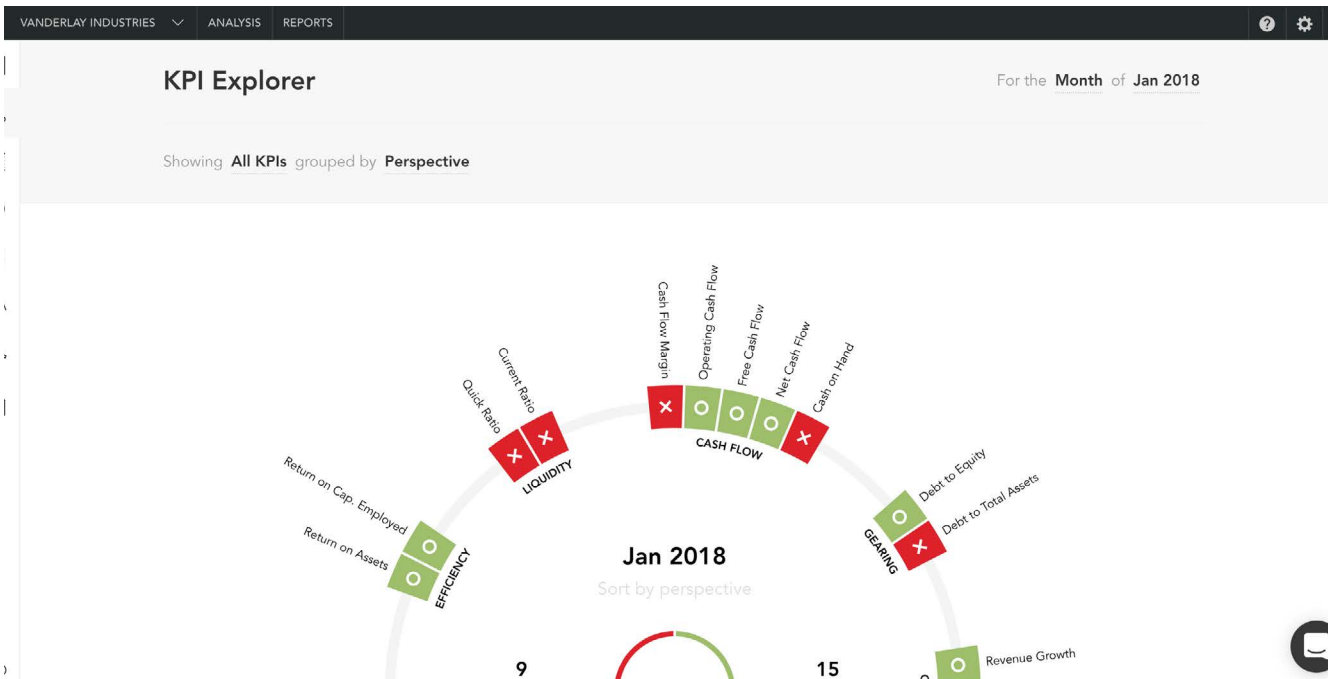
Fathom is a financial analysis tool launched in 2015 by a team from Brisbane, Australia. Using Aurelia and Vue.js, they deliver a useful tool for performing in-depth financial analyses and reporting to 20,000 businesses around the world.

But in 2015, the app encountered several issues.

> *I wasn't content with the performance of the Fathom app. It was written using legacy tools such as old versions of ASP.NET MVC and most notably Adobe Flash, which was due to get deprecated in July 2018, Andrew says.*

The product struggled with scaling, inconsistent architecture, and duplicate code which was difficult to debug and maintain.

> *This was not only having a negative performance impact, it was also making it difficult to evolve the design and architecture of the application. What also concerned us was the risk attached to using old or poorly supported libraries. I definitely saw the need for a prime framework for the app rewrite," Andrew admits.*



There was one more reason for the change.

> ***The intended goal of moving to another framework was to have a dedicated front-end codebase that could be developed, tested, and deployed independently of the main codebase.***

The company explored and seriously considered other frameworks.

> ***I had a look at using Angular 4 which was an obvious option considering my Angular background but I tossed it out pretty quickly. React was appealing as it was growing pretty quickly at that time. To be honest, it was a close call between Vue and React. But what tipped the balance was Vue's syntax and documentation,*** he reveals.

> ***Besides, we were (and still are) using Aurelia as our framework for our reporting product. Aurelia is a great library in itself, but unfortunately it lacked the rich documentation, tools and community support that we as a business require to feel confident in investing in a stable framework for the future.***

Solution

Transitioning to Vue.js was not really part of Fathom's plans for the future.

> ***When I started working at Fathom, the company had no intentions of using Vue to replace the front-end codebase. But I soon realized that the current approach wasn't going to scale as intended and I was confident that Vue would be the right choice for the company going forward," he felt. "Besides, at that point, Vue had been updated to version 2 which was a huge step up. I definitely saw it as a prime candidate for a product rewrite.***

So Andrew started working to prove that.

> *Initially, I began rewriting parts of our product in Vue on my own in my spare work time. When I wasn't working directly on implementing new features nor fixing bugs, I was essentially putting to Vue to create some kind of a demo. I worked on that for about three months and got a few features of our product rewritten in Vue.*

He shared his views with the team.

> *I put the decision to the team but as lead developer, I needed to make the final choice. And we shared a common feeling that React and Angular were generally nice while Vue had a really, really nice and growing community. But it was the documentation that was probably the huge winner for us.*

As there was no Vue expert on the board, the team and the management naturally had some doubts.

> *Two developers were somewhat familiar with Vue.js but the rest have never used it. However, probably the main hesitation was that the framework was back then pretty much maintained by one person, Evan You, plus maybe a few core members. React, on the other hand, had a really strong community and was backed by Facebook, and Angular by Google. So questions about the future arose, whether it's going to be stable in three years' time. So it was quite a gamble.*

He soothed their concerns by presenting a proof of concept for both groups.

> *So I curated a business case for the directors of the company to show the strengths and weaknesses of each framework. Things like API, documentation, community support, etc.," he explains. "I presented the demo*

> *to the management. I also created training materials with some small examples for our developers to follow. Evaluating Vue as a viable solution for rewriting the app was a crucial step. It took a few weeks, lots of research, talks, and business cases for a double check, but it managed to convince them to embrace Vue and we kickstarted the process,* Andrew admits.

Once Vue joined their stack, parallel development began in full swing.

> *When we made that decision, I continued rewriting the app myself for the first few months, while our two prime developers continued implementing new features because we couldn't stop growing as a company nor could we ignore bugs," he says. Then I parceled the work out gradually to other developers who worked on that legacy code. And we also started hiring some more developers so that I could introduce them to the new architecture. It terms of getting up to speed, it all went very, very quickly. It wasn't until early 2018 the rest of the team finally migrated on to the new architecture.*

The migration process went as follows.

> *We adopted Vue and split front-end code into a dedicated single page application repository and wrote new features into the new architecture. We built reusable transition components that allow us to have consistent animations throughout our app and extended Vuex at runtime by dynamically loading extensions to our application (ie. Analysis, Reporting etc). What's more, we used Vuex to create a high performance report editing experience,* Andrew explains.

> *We've written our visualisation library using D3 and then we wrapped D3 components into Vue components so we could create reactive visualisations that emit events.*

Concerning tooling and setup, they used the following:

> ***We used tools like yarn workspaces and webpack to create a modular codebase that is easy to test and code-split. This has the added benefit of reducing code duplication and complexity. We also created a consistent style guide, using Yarn Workspaces, Webpack, vue single file components, and the flex layout attribute library.***

What the team cherished the most about Vue.js were its single file components.

> ***The aspect that we rely on most would be Vue's single file components. They have drastically changed the way in which we can implement atomic design principles. SFC's allow us to have encapsulated logic and consistent styles. Being able to have a template, script and style in one place is great. Other parts we really like about Vue are v-nodes, scoped slots, and state management.***

Vue's flexibility also played an important role.

> ***I have always been committed to pushing Vue to its limits. We tend to bend Vue in ways that it probably shouldn't be bent," Andrew adds with a laugh. "We needed to do some more complex things so I spent a lot time in the source code, so we could get in a little bit deeper. We use scoped slots quite heavily to build really generic and dynamic components.***

Although Vue abounds in multiple features, Andrew noticed a couple of drawbacks.

> ***The one thing that was somewhat irritating, which I believe Evan has fixed in Vue 3.0, is having to put a container element around an array of child***

> *elements at a root template level. It's a little frustrating, but understandable given the complexity around the virtual DOM. Another way we can achieve this is using a functional component which doesn't always work because we need stateful components at certain points of the application. This can get a little bit painful, particularly when there are a lot of components on a page. Having those direct children would be great,* Andrew adds.

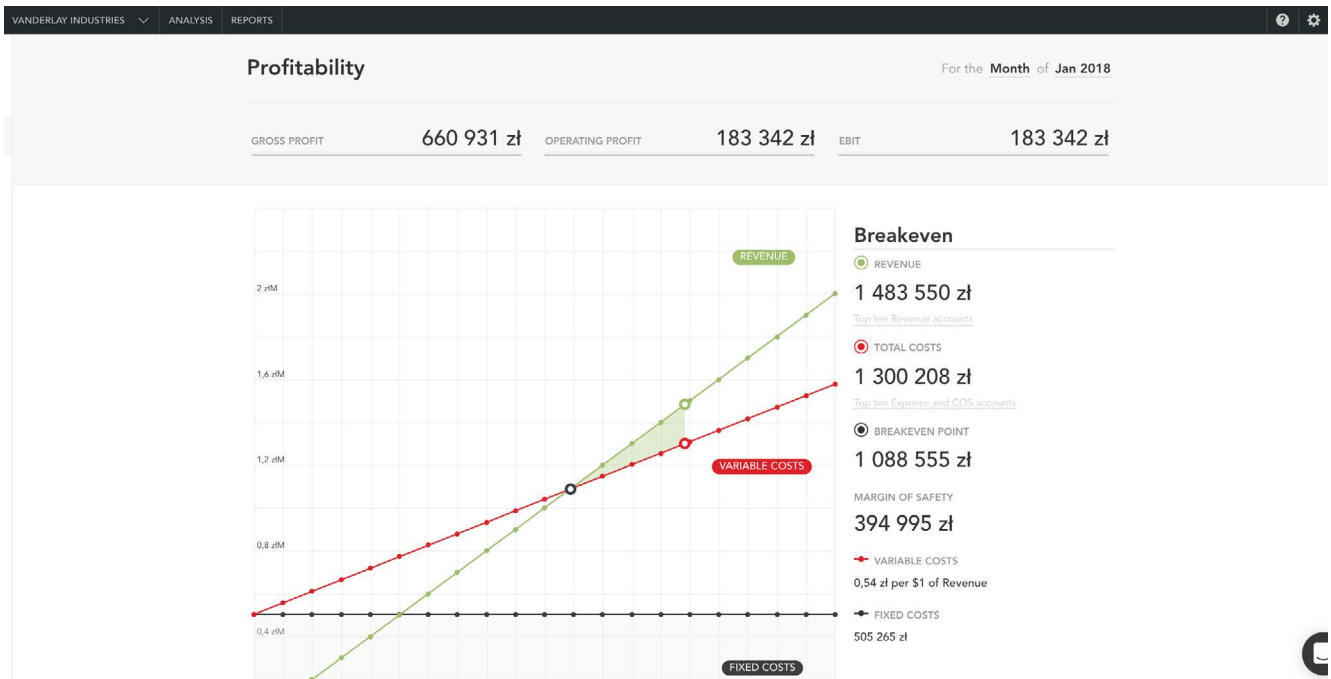
Outcome

The main pain point solved with Vue was successfully completing the re-writing process.

> *Aside from scalability and fixing fragmented design issues, we managed to design, develop, and test a complete rewrite of one of the core features of our product in just 6 months thanks to Vue.*

On the technical side, the improvements are clearly visible.

> *We built a flexible, stable, foundational component library that allows us to build complex interfaces with relative ease. It would have been quite difficult to pull that off before. We also managed to improve debugging and mainte-*



> *nance through the introduction of consistent project structure, architectural patterns. Vue's component structure works really nicely for us in keeping styles and templates really encapsulated. The front-end team can now operate independently with a dedicated repository. This has flow-on effects, including improved response times to critical bugs and a more efficient CI pipeline.*

> *The Codebase is now separated into modular packages that can be easily maintained, tested, and shared across multiple applications. This has also eliminated code duplication. It wouldn't be as easy with any other frame-work as it was with Vue.js.*

Looking at numerical data, the results were quite impressive.

> *Since we implemented Vue, we've been able to cut our app's memory usage by half which is a pretty big change for us. Also, by switching to a strict atomic design structure and using component-driven design, we've been able to cut down our stylesheet size by 67.3%.*

> *Generally, we noticed huge increases in speed, productivity of our users, and usability. Vue helps us save time through its simplicity and great tooling, we have great performance.*

They now deliver a polished, performant, and well-supported product that customers love.

> *When we initially rolled the app out after the first product rewrite with Vue, people's reactions were excellent. They were sending us feedback saying it works way faster. We also gained some nice comments about the refreshed design, saying that it looks, works, and feels better. So the feedback has been really good and a lot of that is due to implementing Vue.*

And developers love to work with the framework.

> ***Developers love Vue.js. They come from various backgrounds, having used jQuery or Angular in the past and getting to use Vue is a breath of fresh air. It's easy for junior developers to use these components prewritten in Vue.js and just build interfaces really quickly.***

Vue also helps Fathom stand out from the competition.

> ***Introducing Vue has allowed us to release major features in short periods of time. We are fixing bugs and implementing new features faster than ever. It's really giving us a bit of an edge on the competition at the moment.***

Fathom is planning to expand the usage of Vue to more products in 2019.

> ***Although the current version of Vue is really feature-rich, we're looking forward to getting and testing Vue 3.0 until it's stable. We're also going to continue implementing Vue in new products we create,*** he shares.

IBM Hybrid Cloud

IBM Hybrid Cloud offers full-stack IT management solutions that spans public, private, and hybrid cloud environments. Dozens of products and services like the IBM Cloud, and Process and API Management are used by thousands of enterprises, across more than twenty industries.

Stephane Rodet
Senior UX Engineer
at IBM Design

Vue.js is the technology that actually helps us just do stuff. When I use Vue, I spend very little time googling things, I can focus on writing the code. Besides, because of its independence, we have guarantee that it will continue to develop and that its licensing is completely safe which is crucial for IT companies like IBM.

CHALLENGE

Wordpress-based platform wasn't able to keep up with the organization growth

Difficulties coordinating work of 200+ designers and co-workers

Lack of transparency between project stakeholders

Outdated design of the website

SOLUTION

Facilitate learning through assigning one person to one technical issue/area

Using Vue, VueX, and animation libraries on the frontend

Relegating the Wordpress to CMS role only

OUTCOME

Coordinating 200+ designers, developers and stakeholders became much easier

The cloud infrastructure makes the website much more flexible

Overhaul recognized by Fast Company's with an Innovation by Design Award in the Web Design category

Challenge

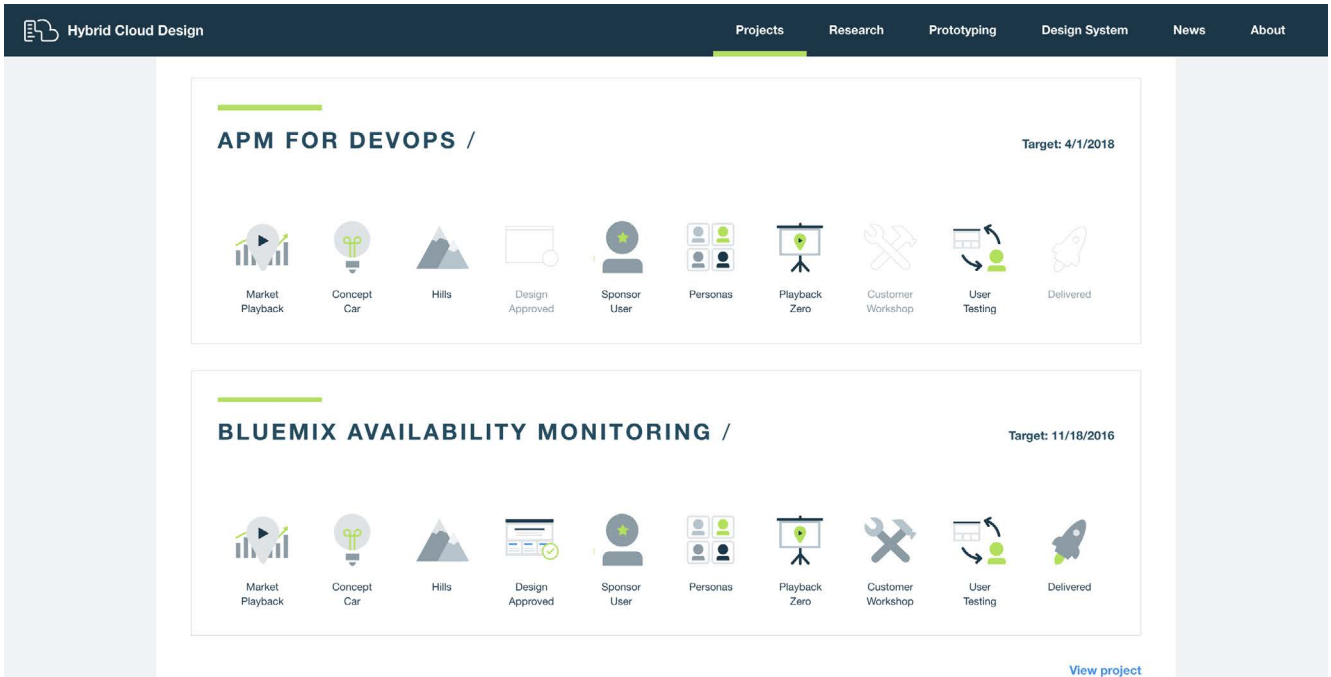
IBM Hybrid Cloud Design is a big organization, with more than 200 designers on board (over 1,000 designers in total), in studios around the world. Its goal is to design IBM Cloud platform and associated products. Keeping their work organized and communication smooth was quite the challenge with the team growing and documentation piling up.

> ***Every project at IBM involves multiple developers, clients, stakeholders, sponsors, etc. They all needed to be in the loop. And it was tough to exchange documents, knowledge and insights while having all of these scattered across different locations.***

The Hybrid Cloud design team dreamed of creating a website to synchronize the design work.

> ***We wanted to build an internal design management system for IBM Hybrid Cloud to have a one single place to share our tools, design processes, updates, and keep everyone on track with what's going in the organization.***

They implemented a solution for that problem.



> ***We already had something based on Wordpress but it was not able to keep up with our growth, and didn't allow for fast coding. So that one question popping up was: 'How can we further expand the existing website and make it usable for a bigger number of people?' That's when we started looking for alternatives,*** he recalls.

A prime requirement was the cloud infrastructure.

> ***We were looking for a Cloud-based infrastructure where we could use methods like continuous delivery and automated staging. We wanted to use a platform-as-a-service kind of cloud to simplify our job, to keep our frontend developers focused on the frontend and not caring about the backend or PHP stuff.***

Other frameworks were considered in the meantime.

> ***Back then, React was already out there, but its learning curve felt too high. Angular wasn't exactly the popular favorite because of its complexity and weight. We even investigated using vanilla Javascript and Web Components. Eventually, we started investigating Vue.js at the end of 2016 and decided to go for it. As it turned out, Vue became a way to satisfy all the interested parties,"*** Stephane laughs.

Solution

The decision to go with Vue dates back to mid-2015, when Stephane first heard about it.

> ***A colleague showed Vue to me, saying that it seemed cool and neat. But I didn't explore the idea right away because at that time I was oriented toward a Web components-kind of structure.***

Deciding on one framework was not an easy decision.

> ***Our work as the UX engineers in the Design organization is to help designers create a system that would be reusable in terms of code. We help the engineers by translating what the designers had in mind. So for a while, we were searching for a fitting technology that would be easy for engineers and designers to understand, and port to their own stack whenever needed,*** he says.

Working towards more cohesion between code and design made Stephane think of Vue again.

> ***When we started transforming this Wordpress-based platform, we instantly thought about Vue. We did the research and had a couple of lengthy discussions in the team to assess if Vue would be working for us. We consulted our manager and actually got him very enthusiastic about Vue.js. We then agreed that this should be the path to take,*** Stephane shares.

What also spoke in favor of Vue was its structure.

> ***Vue also provided us with the ideal kind of component structure we needed. Having CSS, Javascript, HTML in one place enables reusability and that's what brought us to Vue in the first place.***

They started the transition by deviating a bit from some Agile methods.

> ***Due to the number of things that had to be discovered, learned and solved in a short amount of time, we deviated from the Scrum method we used to follow, where you have small stories and a developer works one or two weeks on an end-to-end story. What we did was split the technical work to assign individual jobs to a team member skilled in a specific area. We had***

> *someone dedicated just to implementing and defining SVG animations. I worked on integrating the API, handling the router, and defining data structure. We had a lot of roles and frontend architecture work.*

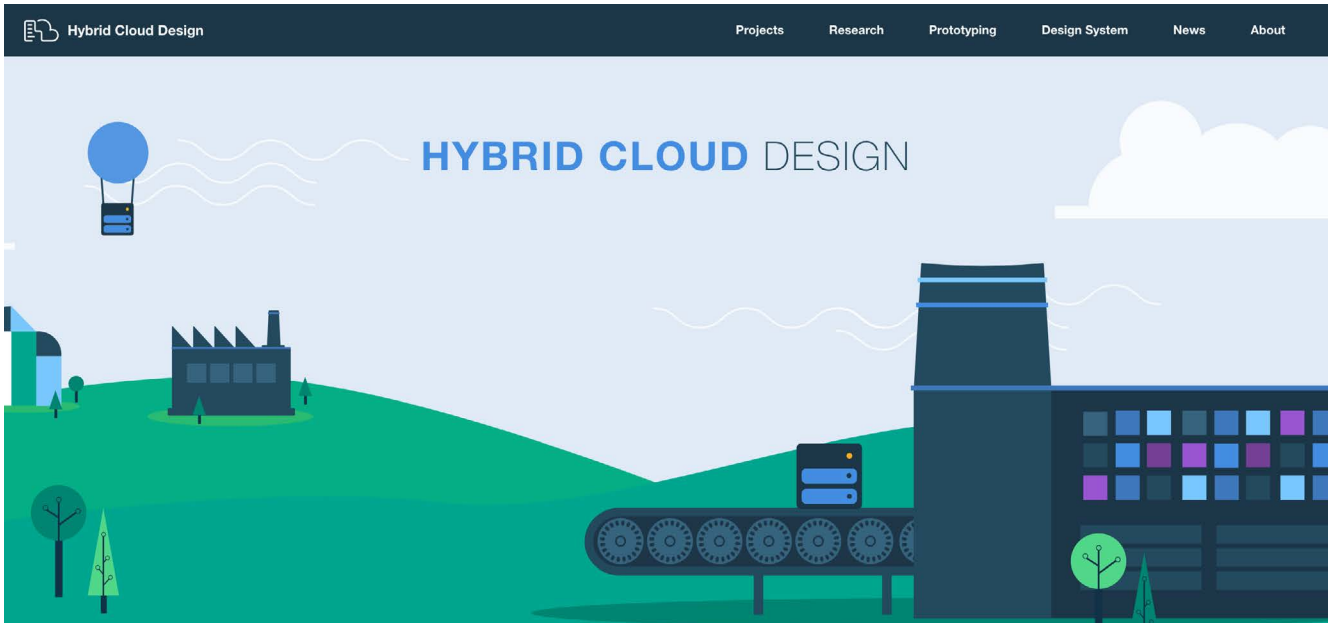
The old team and new hires took up Vue really quickly.

> *New team members usually start making pull requests after a few days. It took us around three, four months to get our new website together working in a basic form. From there on, we could iterate features one after the others.*

> *In our team, we look mostly for people with a great understanding of the web standards: CSS, JavaScript, browser APIs. It was much more important that they knew CSS and how to organize it properly. We didn't have many React or Vue ninjas. We all built up skills over time, as we expect people to learn the framework if needed. And it's possible due to having more experienced people who mentor the new ones,* he explains.

Currently, the setup looks as follows:

> *We built the IBM Hybrid Cloud Design website on top of WordPress by using it as a headless CMS, and with Vue, VueX, and animations libraries on the frontend. We also used an internal component library and Webpack*



> *to build it. We've gone heavy on animations as we wanted to demonstrate that enterprise doesn't mean boring, and we wanted our designers to have fun,* Stephane admits.

Outcome

First and foremost, Vue helped the IBM organization achieve their intended goals.

> *Our team built a design management system for our Hybrid Cloud design organization at IBM, so we can properly coordinate our team of 200+ designers, with their developer teams and their stakeholders, up to senior executives. Moreover, Vue.js helped us separate ourselves from the Word-press-based platform—now we use it only as a CMS that delivers the data and provides a place for authors to edit their content.*

And they're seeing benefits of the transition on a daily basis.

> *We stay focused on our design work, educate our stakeholders about our design practice, and showcase the impact and benefits that design can bring to the overall success of products and offerings. The site is used on a daily basis by various types of users, including designers and the organiza-tion's user researchers, as well as engineering, product management, and executive stakeholders.*

In general, Vue.js made everything simpler and faster.

> *For us, as the design organization, it's quite natural to use much more of CSS than Javascript. So having Vue.js on board made our cooperation easier—it feels much like native HTML while providing a lot of additional value.*

The project turned out to be such a success in terms of design and mission that it began garnering acclaim even outside the IBM organization.

> ***The Hybrid Cloud Design website was awarded an Honorable Mention by Fast Company's Innovation by Design Awards in the Web Design category. The Innovation by Design Award recognizes outstanding creative work at the intersection of design, business, and innovation,*** Stephane shares.

> ***Indeed, the website functions as an educational resource, too, raising awareness about the design process, the team's various deliverables, and the design disciplines within the organization. It's also a management system for our team to track the execution of the various design initiatives and to share project work with each other and our various stakeholders.***

Vue also works as a recruitment incentive now.

> ***We see positive reactions when we tell candidates that we're using Vue. And we're still seeing people interested in Vue in the organization who really grow to enjoy it. Developer satisfaction is very high, we often hear many devs wishing that more frameworks were similar to Vue.js. But once they get to know it, they tend to stick to it.***

Even though Vue.js was accepted and warmly welcomed in the organization and brought visible benefits, Stephane still sees some drawbacks.

> ***Developers within the company often choose React because of its wide usage and popularity, especially for bigger projects. But at the same time, I see many people who like Vue.js and would want to use it more. So what is currently missing is not the Vue ecosystem itself. Rather, a few libraries and templates, mostly connected to our design standards,***

> *are missing from our internal ecosystem. Once we have that, adopting Vue will be much easier.*

On the technical side, Stephane emphasizes one thing.

> *There are some things that Vue.js slightly lacks and some things that are hard for people to grasp, e.g organizing (sometimes complex) code through many components in big applications. But I'm really looking forward the upcoming Vue 3.0 release with class components which I find a major improvement."*

> *Another remaining issue is the dependency management, the whole npm and Webpack side of things. Before we ship things outside, we are required to show proper due diligence of the libraries with their licensing, etc. Consequently, the more dependencies you have in your package.json, the more issues you have to discuss and resolve. This is always a bit of a problem but it's not only specific to Vue.*

> *Concerning Webpack, the configuration can be very daunting so I think that Vue CLI solves a lot of that.*

The advice Stephane would give to other enterprise—level organization is quite straightforward.

> *I think you should go for Vue. It offers an open and independent community, not backed by a company that would give it directions and potentially serve as your competition at the same time. Companies have a guarantee that it will continue to develop, that its licensing is completely safe which is a major consideration for IT companies. Vue is well-documented, easy to learn, and lightweight so it helps performance and reduces hosting costs.*

Laravel

Taylor Otwell
Founder of Laravel

Laravel is a free, open-source PHP framework that makes development a delightful and truly fulfilling experience.

Vue let us create the most productive combination for building Web applications on the frontend right now. Actually, it's the fastest way to start to building applications. If it hadn't been for Vue in the first place, we wouldn't have released a frontend integration with Laravel at all.

CHALLENGE	SOLUTION	OUTCOME
Providing developers with a seamless experience	Benchmarking other frameworks like React	The most productive combination for building Web applications
Difficulty scaling the app	Approving the integration by Evan You, Vue.js creator	Simple and powerful option for the frontend
Ambitious plans for implementing integration in new products	Pairing Laravel and Vue in a subscription software app	Positive feedback from developers

Challenge

Laravel is an open-source PHP framework used by over 100,000 developers worldwide. Since its launch in 2011, it has been installed 52M times and currently averages as many as 50-60,000 downloads every day. But back in 2015, when Taylor started building Laravel Spark, it turned out that their stack was missing a highly performant frontend framework.

> ***Laravel Spark was a starting point for a subscription software and in the case of such apps you gain hundreds of users monthly. We needed a technology that would scale up to much more complicated applications in the future but remain simple.***

The varied audience turned out to be Laravel's biggest concern.

> ***I knew that thousands of people were gonna use it, people with different levels of programming skills. Some of them would be beginners, others would have been out of the game for like 20 years. It's always the issue we're facing with such a wide audience,*** Taylor explains.

Another challenge they faced was finding a solution that would be really simple but powerful at the same time.

> ***I wanted to make something really robust but also simple so that advanced programmers could fiddle with it and newbies could catch on easily. As I'm not really a JavaScript expert myself but more of a PHP programmer, I was looking for a framework to pick up and use almost in no time. And it wasn't the most obvious feature of the frameworks out there," Taylor reflects. "On the technical side, I needed the framework to ship logs and make screens out of the box really smoothly.***

In the meantime, the team considered other frameworks as well.

> *Other frameworks might be powerful but not simple, or are simple but don't have specific features. When deciding which framework to integrate with, we considered React.js, but it was more complicated and would be more time-consuming. The syntax felt different as well and would be a lot harder to learn.*

.Solution

When other frameworks failed, the choice narrowed down to Vue.
js and there was not much hesitation.

> *What I liked about Vue is that our stories are similar. Both frameworks started from being a one-man project, expanded to a commercial scale due to a shared philosophy of being easy to learn and scale," Taylor adds. "Vue.js is used by lots of developers, there are lots of tutorials out there and its syntax is so easy to build with. It's easy to get feedback from the community as well. Laravel is the same way.*

> *I had a lot of discussions with Evan You, Vue.js creator, and Jeffrey Wey who's the creator of Laracast.com. We all agreed that making Vue.js a default framework for Laravel was the best answer to Web developers' needs."*

The biggest advantage of Vue.js that Taylor mentioned was its consistent simplicity.

> *With Vue.js, it's really simple to read the whole documentation and the model binding data is easy. The helpers for looping through the data, like HTML helpers, allowing the user to customize, are gold. All those char-*

> *acteristics are much nicer than what any other framework offers. It made it all a really easy choice,"* he says.

Having made their choice, the team sat down to work.

> *We paired Laravel with Vue.js to build a whole product, Laravel Spark, a subscription software like Gitlab or Codeship. It contained all code, frontend, and billing screens where you could download your invoices. And the whole process with Vue was really nice because we didn't have to add any compilation steps to the Laravel installation process. We could just include the Vue pile. And we didn't need to worry like about React compiling JSX or any other type of weird syntax,* Taylor recalls.

Outcome

The integration brought more benefits to Laravel that the team had anticipated.

> *To be honest, Vue.js was so polished that I never had to ask for any features. It has it all. It helped us do pretty much what we wanted to do, pretty well. Vue and Laravel turned out to be a really magic formula that we needed at the time. Talking about power and simplicity, that's where both Laravel and Vue shine, a lot. And merging those aspects can be really done only with Vue.js.*

And Vue showed its power through the features.

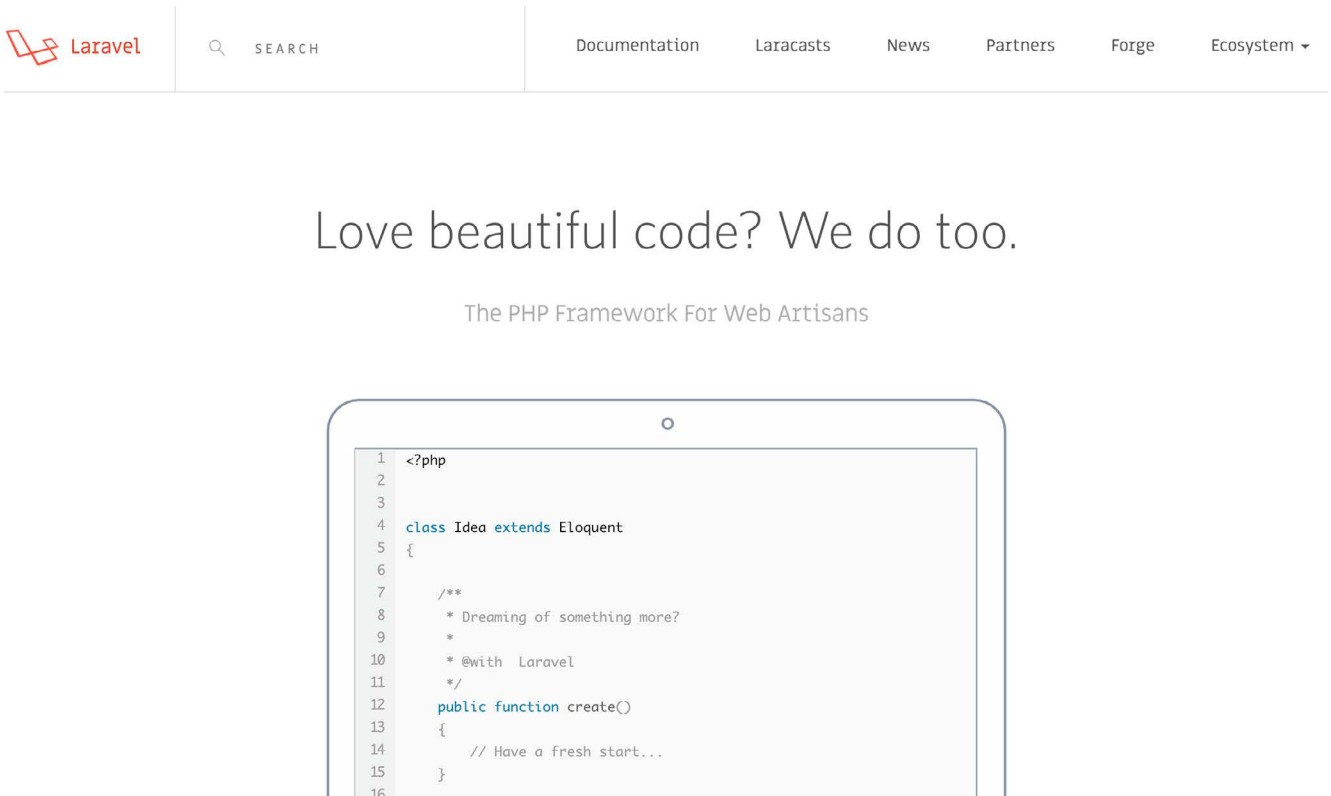
> *The most powerful aspect of Vue is a cohesive, integrated ecosystem. The combination of Vue itself, Vue Router, Vuex, Vue Debugger DevTool, etc. all come together to form a really productive environment. In addition,*

> *the quality of the tools is very consistent because they are all maintained by the Vue core team which has a high standard of quality.*

Vue.js helps Laravel build apps really fast and smooth on the frontend, which contributed to Laravel's overall success.

> *As a result, Laravel Spark was a commercial product sold in over 7,000 licenses and Laravel Nova has sold over 5,000 licenses. Laravel Horizon has been downloaded 1.2 million time and Laravel Telescope, released only in October 2018 has already reached 100,000 downloads.That's a pretty big win for our company. And we wouldn't really have that if it hadn't been for Vue.js features.*

> *People buy it, download it and once it gets set up, they have access to the code. They can modify it but it's all kind of prebuilt so they have user management access. It's very comfortable and attracts a lot of devs. Many of them have been using it to build their apps since 2015 and Vue.js integration made it a lot easier to work with in a custom manner. Engineers who work with the Laravel+Vue combo convey only positive feedback and look forward to more integrated products.*



If asked in 2019, the Laravel team would still have made the same decision.

> ***Three years later and we still love Vue.js. We only use a lot more components like Vuex and other features. The whole time we've been working with Vue, the quality remained the same supreme level.***

Taylor is planning even more integrations with Vue.js.

> ***I'm working on Laravel Vapor right now. Hopefully I will release it next summer at Laracon. It helps people deploy the Web app really easily, it sets all your hosting and SSL certificates up. Basically, with Laravel, I try to build a kind of a story from beginning to end," he continues. "You develop it on a new machine and then I try to provide a nice way to deploy it when you're done. It will be a successor, an improved version of Laravel Forge, which is my main commercial product.***

The Future of Vue.js

Evan You
Creator of Vue.js

Technical Evolution

Vue.js grew into the framework it is today thanks to the its community. The evolution of Vue.js is fundamentally driven by the interest and needs of its users. We learn from the problems our users are trying to solve, and our ultimate goal is to help you solve these problems in a faster and easier fashion.

To achieve that goal, it requires Vue.js to constantly evolve to take advantage of the new emerging platform features and community patterns. There are a lot of exciting things going on in the frontend world, and you certainly run into them every single day. Here we will outline what some of the most important new ideas and what they mean for Vue.

ES2015+

In 2019, we are now seeing much more mature native support for ES2015 and above in mainstream browsers. Most evergreen browsers now have full support for ES2015, and are generally very fast in implementing new additions to the language. As older browsers phase out, we will eventually

be able to take full advantage of these new language features. For Vue, there are multiple areas where we will benefit from it:

- **Core**: Vue 3 will be able to leverage native Proxies for more performant change detection and fewer edge cases. Class fields getting closer to stage 4 also means we can provide a native class-based API that works nicely for both plain ES and TypeScript. Finally, shipping Vue itself in native ES2015+ also results in smaller file size and better performance.
- **CLI**: Vue CLI 3 already provides a "modern mode" that allows you to ship un-transpiled ES2015 code to browsers with native support, while shipping a separate bundle for legacy browsers. This results in smaller bundles and better performance for users on modern browsers.
- **Vuex**: with async/await becoming the new standard way of handling async logic in JavaScript, we are planning to re-design Vuex's API so that it is no longer necessary to separate actions and mutations just for the sake of handling async logic.
- **Router**: Vue Router already works very well with code-splitting via webpack, using the dynamic import syntax as a compile-time hint. But it also works equally well with native dynamic imports, as Vue's async component loading logic is decoupled from the underlying implementation.

TypeScript

TypeScript's popularity has grown significantly in the past year, especially in larger scale projects. This is partly due to TypeScript itself being a solid language paired with great IDE support, and partly due to the rising amount of large-scale projects in the frontend. Although writing code with types makes it less flexible upfront, it pays dividends in long term maintenance, especially during refactors. We do not think TypeScript is necessary

in every project, but we do acknowledge its importance for those that find it beneficial to their workflows.

Vue's current TypeScript integration story still has a lot of room for improvement, and we are planning to tackle them in 3.0. The new major version will provide a native class-based API for authoring components, TSX support, and in general more solid type definitions due to us switching to TypeScript as the internal implementation language. We are also exploring the possibility to provide type-based intellisense in Vue single-file component templates with help from the VSCode team.

One important thing for non-TypeScript users to note is that being TypeScript-friendly or using TypeScript for Vue's own implementation doesn't in any way mean we have to compromise the development experience for those that do not use it. Plain JavaScript usage is always going to be our baseline. We will always make sure to keep the base API simple for a smooth learning curve, and will never force you to go with TypeScript if you don't want to.

Web Components

We often see articles suggesting that native web components will be able to replace existing frameworks in the future - but that is not the case. The finalized web components standards only provide very low level primitives for defining custom elements with its own isolated (shadow) DOM. They do not provide the necessary structure, patterns and tooling needed to build a real application. Instead, you will still need a framework to fill these gaps even if you are using native web components.

A promising use case for web components is for encapsulating a component and shipping it as a framework agnostic, native custom element that can be used anywhere. This is what enables mobile solutions like Ionic 4 and Onsen UI to support multiple frameworks at the same time. Vue CLI 3 also provides the ability to build and distribute your Vue components as native custom elements.

Another aspect of web components that we intend to investigate is leveraging Shadow DOM as a native mechanism for CSS encapsulation. However, at the moment of this writing, Shadow DOM native support is still unsatisfactory, and the inability to declare it directly in HTML makes it incompatible with server-side rendering.

HTML Modules

HTML Modules is a new proposal under the web components umbrella that is more relevant to Vue.js as it is very similar to Vue's SFC (single-file components) format. It is currently still being discussed, but the Edge team from Microsoft has posted an intent to implement in the Chromium project.

HTML Modules allows users to natively import HTML documents as a module in their JavaScript. Each HTML document can in turn export a JavaScript module together with an HTML template and styling. This would bring the development experience very close to what Vue developers are used to today. We are keeping a close eye on it and are involved in the feedback process. We hope that HTML Modules will enable Vue to provide a near-native workflow for single-file components, which will be great for small to medium scale projects.

WebAssembly

WebAssembly provides the ability to perform heavy computations in the browser with significant performance advantage over regular JavaScript. This is great for infrequent, intensive computations, such as image or video processing. However, a runtime framework like Vue.js performs very constant updates and needs to perform DOM mutations.

Since the WebAssembly worker has no access to the DOM, it would have to constantly communicate back and forth with the main thread, and the cost of such communication likely outweighs the potential performance gains, or at least offsets it so much that it no longer justifies the extra complexity. On the other hand, it could also be challenging to simultaneously target browsers with different levels of WebAssembly support.

One area where we think it might be more promising to leverage WebAssembly in Vue is server-side rendering. Due to the fact that Vue projects primarily use templates, it is possible to pre-compile the templates and render them via WebAssembly for significantly better performance.

Mobile Apps

Vue 2.0 was designed to decouple the core runtime with platform-specific operations, so it already supports rendering to non-web targets. There are projects like NativeScript Vue and Weex that allows users to build truly native applications using Vue's syntax and API. In China, there are also multiple solutions that leverage Vue's such ability to target multiple "mini program" platforms at the same time.

However, Vue 2.x lacks a first-class API for building custom renderers. This requires the authors of the solutions mentioned above to fork Vue's source code and creates a lot of maintenance complexity.

In 3.0, Vue will provide a first-class custom renderer API. Users can use Vue's core runtime as a dependency and build on top of it. The API will allow easy bridging to iOS/Android native renderers, or other use cases such as testing or rendering to terminals.

For lighter-weight use cases, it is also worth considering cordova-based solutions such as Quasar Framework, Ionic 4, or Onsen UI, all of which have been around for a long time and fairly mature.

New Patterns and Ideas

We are always keeping an eye out for emerging new patterns in the larger frontend ecosystem and thinking about how Vue can learn from them. Some of the recent prominent examples include React's new Hooks API, new capabilities enabled by its new Fiber architecture, and the compiler-centric, minimal-runtime design found in Svelte and Angular's Ivy compiler. We have been investigating how these new ideas can benefit Vue users without deviating from Vue's own idiomatic usage, and we have made some good progress in the 3.0 prototypes. We will share more details once we figure how they will affect the public API design.

Growing the Project

An open source project is more than just the code. It's also the people, the process and the ecosystem around it. As the project grows, we are learn-

ing along the way. Here are the things we are working on in 2019 to ensure Vue is on track for the long run:

RFC Process

To truly understand the needs of our users and help them, the users need to have a voice in the design process. As a community-driven project, it is important for the community to take part in shaping the future of Vue. This is why we adopted an open RFC (Request for Comments) process. The process is required for landing any future substantial changes in the framework, and forces each new proposal to provide thorough considerations of alternatives, potential drawbacks, and its impact on existing users. This ensures the long term stability of the framework. The RFC process is also conducted in the open, so every community member can participate by providing feedback or submitting their own proposals. We have already landed a few changes by going through this process in 2.6.

Operation and Contribution Guidelines

We have started an Operation Guidelines Handbook which will expand to cover all aspects of Vue's operation as a project. This includes our governance model, contribution workflows, issue triaging guidelines, release process etc. Most importantly, it will also include guidance on contributing to Vue, both in technical or non-technical forms. For 3.0 we also plan to provide detailed technical explainers for the internal architecture so that more users can land PRs in the codebase. By lowering the barrier of contribution, we hope that we can further leverage the power of the community.

Sustainability

As the creator of Vue.js, this is the third year I have been working full time on Vue.js, with consistently growing financial support from Patreon and various other sources. I hired my first full-time employee in 2018, who has been primarily funded by our OpenCollective donations. Financially, aggressive growth is not our goal. We aim to find a balance where we can be sustainable for the long run while being able to focus on the project itself instead of running a business.

We are also seeing more community members doing full-time consulting exclusively on Vue projects, or building profitable businesses by offering Vue-related content or services - both good signs for the sustainability of the greater Vue ecosystem.

Reliability

As Vue's adoption keeps growing, more and more developers want to use Vue in their company projects. We understand that picking a framework to use isn't always a purely technical decision, and we want to make sure that Vue is a suitable choice for businesses of any size. In 2019, we plan to further improve Vue's story for enterprise usage:

More robust and predictable release cycles: with 2.6 shipped, we plan to transition into a 3-month minor release cycle so users can better prepare for updates.

Regression testing system: we already have a working regression testing system that helps us test compatibility with major libraries

before release. We plan to further expand this to include more libraries, and eventually provide a workflow for users to subscribe and run custom tests on nightly builds.

We are investigating the possibility of providing commercial support in the future. This is targeted towards companies that have policies or concerns regarding adopting software without paid SLAs.

Conclusion

We've worked really hard to make Vue one of the most approachable, reliable and well-rounded solutions for building web applications today, and with the support from the community, we are confident we can keep it that way for a long time to come. We are excited to grow the project together with you.

© Monterail, February 2019

Monterail is a close-knit team of 80+ experts
offering Web & mobile development
for startups and businesses.
And we kinda love Vue.

www.monterail.com
hello@monterail.com